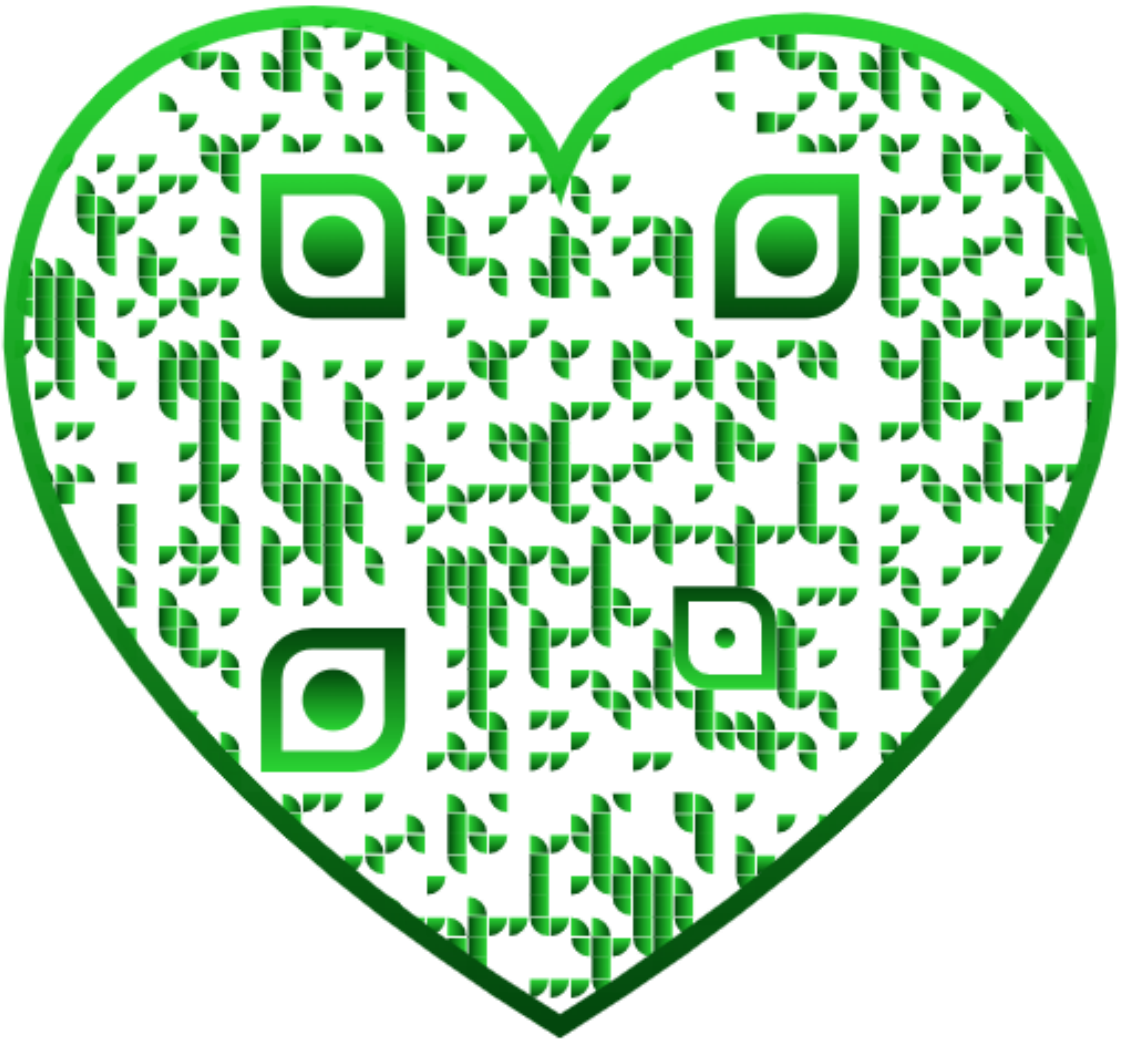
Avec les Nuls, tout devient facile !

2e édition

Raspberry Pi pour les nuls

- Connecter et configurer le Raspberry Pi
- Installer et explorer Linux
- Travail et détente avec votre Raspberry
- Programmer avec Scratch et Python
- Construire ses premiers circuits électroniques

Sean McManus
Mike Cook



Raspberry Pi

pour les nuls

3e édition

Sean McManus
Mike Cook

FIRST INTERACTIVE

Raspberry Pi pour les Nuls, 3e édition

Titre de l'édition originale : *Raspberry Pi For Dummies, 3rd Edition*
Copyright © 2017 Wiley Publishing, Inc.

Pour les Nuls est une marque déposée de Wiley Publishing, Inc.
For Dummies est une marque déposée de Wiley Publishing, Inc.

Édition française publiée an accord avec Wiley Publishing, Inc.
© Éditions First, un département d'Édi8, Paris, 2017. Publié en accord avec Wiley Publishing, Inc.

Éditions First, un département d'Édi8
12, avenue d'Italie
75013 Paris – France
Tél. : 01 44 16 09 00
Fax : 01 44 16 09 01
Courriel : firstinfo@efirst.com
Site Internet : www.pourlesnuls.fr

ISBN : 978-2-412-03443-9
ISBN numérique : 9782412036587
Dépôt légal : février 2018

Traduction de l'anglais : Olivier Engler
Mise en page : Enredos e Legendas Unip. Lda

Cette œuvre est protégée par le droit d'auteur et strictement réservée à l'usage privé du client. Toute reproduction ou diffusion au profit de tiers, à titre gratuit ou onéreux, de tout ou partie de cette œuvre est strictement interdite et constitue une contrefaçon prévue par les articles L 335-2 et suivants du Code de la propriété intellectuelle. L'éditeur se réserve le droit de poursuivre toute atteinte à ses droits de propriété intellectuelle devant les juridictions civiles ou pénales.

Ce livre numérique a été converti initialement au format EPUB par Isako www.isako.com à partir de l'édition papier du même ouvrage.

Introduction

Depuis quelques années, l'enseignement de l'informatique s'est essentiellement consacré à l'acquisition de compétences pratiques de type secrétariat ou bureautique. La compréhension du fonctionnement d'un ordinateur et l'apprentissage de la programmation ont été perdus de vue. Le Raspberry Pi propose d'inverser cette tendance. Vous pouvez vous en servir pour jouer, écouter de la musique, retoucher des photos, et saisir du texte, comme n'importe quel ordinateur personnel. Mais il donne accès à bien d'autres activités, car il constitue une voie d'accès à la programmation, à l'électronique et au monde mystérieux de Linux, cette puissante alternative gratuite aux systèmes Windows et macOS.

Le Raspberry Pi propose de nouvelles opportunités à tout un chacun, mais il peut paraître déstabilisant. Il se présente sous la forme d'un petit circuit imprimé. Pour pouvoir en faire quelque chose d'utile, il faut au minimum y ajouter un système d'exploitation (sur une carte mémoire SD) et le relier à un écran, un clavier et une souris. Pour commencer, vous pouvez soit apprendre les rudiments des commandes Linux, soit accéder à l'interface graphique fenêtrée LXDE. Vous êtes peut-être un mordu qui adore se délecter de nouvelles technologies, ou un utilisateur traditionnel qui a besoin d'un nouvel ordinateur à la maison pour ses enfants. Dans tous les cas, ce livre va vous aider à faire vos premiers pas avec votre Raspberry Pi et vous montrer quelques-unes de ses nombreuses utilisations amusantes et enthousiasmantes.

À propos de ce livre

Raspberry Pi pour les Nuls propose une introduction claire et compacte à la terminologie, la technologie et les techniques qui vous seront utiles pour bien exploiter le Raspberry. Vous allez être guidé dans les opérations suivantes :

- Branchements de votre Raspberry Pi.

- Personnalisation de la configuration selon vos besoins.
- Sélection et installation de logiciels gratuits fonctionnant sur le Raspberry Pi.
- Utilisation de l'environnement graphique pour lancer les programmes, gérer vos fichiers, naviguer sur le Web, visualiser vos photos, *etc*.
- Apprentissage de la ligne de commande Linux pour gérer le Raspberry Pi et ses fichiers système.
- Utilisation du Raspberry Pi comme outil de productivité bureautique et de retouche de photos.
- Création d'un centre multimédia de lecture de musique et de vidéos.
- Composition de musique électronique.
- Création d'animations et de jeux d'arcade avec le langage de programmation pédagogique Scratch.
- Conception de jeux plus évolués avec le langage de programmation général Python.
- Initiation à l'électronique avec une introduction à la soudure de composants, à la conception et la réalisation de circuits électroniques et de jeux contrôlés par le Raspberry Pi.

En quoi ce livre vous sera utile ?

Une fois que vous avez extirpé le circuit Raspberry Pi de sa pochette anti-électricité statique, que faire ensuite ?

Ce livre répond à cette question. Grâce à lui, vous allez au plus vite pouvoir vous servir du Raspberry Pi et découvrir une partie des choses formidables qu'il permet de faire, notamment des projets très concrets : construction d'un site Web, création de jeux vidéo, construction de gadgets électroniques, tout cela sans connaissances préalables, mais avec l'aide de ce livre.

Le Raspberry Pi est un ordinateur un peu différent de ceux que vous avez utilisés jusqu'ici. Ce livre prend soin de vous montrer comment réaliser avec lui les mêmes activités que celles auxquelles vous vous attendez sur n'importe quel ordinateur, comme la lecture de musique et la création de documents.

Vous pourriez bien sûr acquérir la plupart de ces savoir-faire par tâtonnements, mais c'est une technique demandant beaucoup de temps, ce qui peut engendrer de la frustration. Ce livre va vous permettre de prendre en main plus rapidement votre Raspberry Pi, quelle que soit l'utilisation que vous envisagez par la suite.

Quelques conditions préalables

Raspberry Pi pour les Nuls est évidemment destiné aux débutants, c'est-à-dire des personnes qui n'ont encore jamais utilisé ce genre d'ordinateur. Nous avons néanmoins bâti le livre en supposant quelques acquis préalables, car nous n'aurions pas assez de place pour décrire tous les projets captivants s'il avait d'abord fallu rappeler comment utiliser une souris ! Voici nos présupposés :

- **Vous savez déjà utiliser un ordinateur** sous Windows ou sous Apple macOS. Le couple clavier/souris, les fenêtres et les icônes ne vous sont pas inconnus. Vous savez utiliser un ordinateur pour naviguer sur le Web ou rédiger un courrier.
- **Le Raspberry Pi n'est pas votre seul ordinateur** ou bien vous avez accès, surtout au départ, à un autre ordinateur sur lequel vous pourrez préparer la carte mémoire SD qui va contenir le système d'exploitation du Raspberry (Chapitre 2).

- Au niveau des interconnexions réseau, nous supposons que **vous disposez d'une connexion à Internet** et qu'il reste un port réseau libre sur le routeur ou la box. Prévoyez aussi un câble réseau pour le Raspberry Pi.
- **Le Raspberry Pi est votre premier ordinateur sous Linux**. Aucune notion préalable de ce système n'est requise, mais si vous êtes déjà un ninja de Linux, vous trouverez dans ce livre une bonne référence sur la variante de Linux utilisée par le Raspberry Pi.
- **Vous êtes prêt à partager notre enthousiasme** pour le monde de possibilités que le Raspberry Pi va vous procurer !

Ces préalables étant posés, nous espérons que ce livre sera vraiment accessible à tout le monde. Rappelons que le Raspberry Pi est adopté en masse dans les écoles et clubs d'informatique pour juniors ; ce livre constituera une ressource pratique pour les enseignants comme pour les élèves. Par ailleurs, le Raspberry Pi se répand dans les foyers, des personnes de tous âges l'adoptant pour leurs activités éducatives comme pour leurs loisirs (le circuit peut servir de centre multimédia de salon).

Icônes utilisées

Si vous avez déjà lu un livre de cette collection « Pour les Nuls », vous savez que nous y utilisons des icônes en marge pour désigner des informations capitales ou bien complémentaires. Dans ce livre, nous utilisons quatre icônes :

Cette icône désigne une remarque d'expert ou une idée pour parvenir à vos fins avec moins d'efforts.

Tout le livre est orienté technique, mais cette icône singularise un passage encore plus technique. Nous avons fait notre possible pour éviter d'employer un jargon technique et des présentations trop complexes, mais il faut parfois absorber un principe de base pour mieux comprendre ce que

vous faites. Il nous faut parfois adopter un discours plus technique lorsque le sujet le réclame. Les sections marquées par cette icône pourront être lues plusieurs fois pour être bien comprises, mais vous pouvez à tout moment décider que vous n'avez pas besoin de plonger dans de tels détails, et donc ignorer ces passages. C'est vous qui voyez !

Nous aimons croire que la lecture de ce livre sera pour vous une expérience inoubliable, mais au cas où, nous avons prévu quelques rappels. Soit ce sont des concepts importants à mémoriser, soit des points fondamentaux du sujet en cours de présentation.

Comme quand vous conduisez, vous devez ralentir quand vous voyez un panneau d'avertissement. Il signale une zone à risques.

Le fichier archive des exemples

Ce livre contient plusieurs fichiers de code source pour les programmes des Parties IV et V, ainsi que d'autres fichiers utiles. L'ensemble a été compressé et rassemblé dans un fichier archive au format ZIP. Nous vous invitons à le télécharger après avoir lu le Chapitre 4 qui explique comment gérer les fichiers sous Linux dans l'environnement graphique.

Voici comment trouver le gisement officiel de ce fichier archive :

1. **Dans votre navigateur, rendez-vous à la page suivante :** `www.pourlesnuls.fr`
2. **Cliquez sur l'icône de loupe**.
3. **Saisissez « Raspberry » dans la zone de recherche qui s'affiche, puis appuyez sur la touche OK pour valider**.
4. **Cherchez la couverture de ce livre et cliquez dessus**.
5. **Vous arrivez à la page contenant le lien pour télécharger le fichier archive. Cliquez sur le bouton Télécharger**.
6. **Enregistrez le fichier dans un sous-dossier de votre dossier personnel sur le Raspberry et**

décompressez-le.

Notez que vous pouvez trouver au même endroit d'éventuels fichiers de correctifs. N'hésitez pas à revenir consulter cette page si vous détectez dans le texte quelque chose qui vous étonne.

Les anglophones peuvent aller consulter le site des deux auteurs (britanniques) de ce livre : www.thebox.myzen.co.uk et www.sean.co.uk.

Par où commencer ?

Vous choisissez comment vous voulez lire le livre. Il a été prévu pour vous guider pas à pas de la découverte du Raspberry Pi à l'apprentissage de deux langages de programmation afin de savoir créer vos programmes puis à la création de plusieurs montages électroniques pilotés par ces langages. Certains chapitres réclament les connaissances des chapitres antérieurs, surtout à partir de la Partie IV.

Nous comprenons que certains sujets vous intéressent plus que d'autres et que vous aurez besoin de plus d'aide dans certains domaines. Lorsqu'un chapitre réclame des savoirs présentés dans un autre, nous avons prévu des renvois pour en retrouver la description. Nous avons de même ajouté des renvois en avant lorsqu'une solution rapide est présente dans un chapitre ultérieur.

Si vous n'avez pas encore utilisé votre Raspberry Pi, commencez avec la Partie I. S'il est déjà fonctionnel, passez à la Partie II qui montre comment rapidement prendre en mains les principaux logiciels. La Partie III aborde les outils de productivité, de création et de divertissement. Pour plonger dans la programmation, attaquez la Partie IV qui présente les langages Scratch et Python. Vous aurez besoin du savoir-faire acquis en Python pour réaliser les montages électroniques de la Partie V.

PARTIE 1
Premiers pas avec le Raspberry Pi

DANS CETTE PARTIE :

- Découverte du Raspberry Pi et des autres équipements qu'il faut réunir pour s'en servir.
- Téléchargement du système Linux et implantation d'une carte mémoire SD.
- Connexion du Raspberry Pi à l'alimentation, à un concentrateur (hub) USB, à un clavier, une souris et un écran.
- Utilisation de raspi-config pour régler les paramètres de votre Raspberry Pi.

Chapitre 1

Présentation du Raspberry Pi

DANS CE CHAPITRE :

- **Découvrir le Raspberry Pi**
- **Apprendre ce qu'il est possible de faire avec un Raspberry Pi**
- **Découvrir quelles sont ses limites**
- **Se procurer un Raspberry Pi**
- **Faire l'inventaire de ce qu'il vous faut**

Le Raspberry Pi est peut-être l'ordinateur le plus enthousiasmant de nos jours. Alors que la plupart des équipements informatiques que nous utilisons au quotidien (nos téléphones, nos tablettes, nos consoles de jeux) sont conçus pour nous décourager de tenter de les comprendre et de les démonter, c'est exactement l'inverse avec le Raspberry Pi. Dès que vous voyez son petit circuit vert, vous êtes invité à l'étudier, à jouer avec et à créer. Il est livré avec tous les outils qui vous permettront de créer vos logiciels (de programmer) et de contrôler vos montages électroniques. Il est suffisamment bon marché pour vous libérer de toute crainte de le casser en le découvrant ; sa destruction éventuelle ne risque pas de mettre votre compte bancaire à découvert. Vous pouvez tout essayer en toute confiance.

De plus en plus de gens sont emballés par son potentiel et découvrent sans cesse de nouvelles manières de l'utiliser. Un certain Dave Akerman (www.daveakerman.com) a par exemple, avec ses amis, attaché un RasPi (son petit nom) à un ballon-sonde qu'il a ensuite envoyé dans la stratosphère à 40 km d'altitude pour prendre des images de notre belle planète avec une webcam.

Le professeur Simon Cox et son équipe de l'université de Southampton ont interconnecté 64 circuits Raspberry Pi pour constituer un

supercalculateur expérimental, maintenu en place par des briques Lego. Dans ce supercalculateur (Figure 1.1), tous les Raspberry Pi travaillent ensemble pour résoudre des calculs complexes. Le projet a permis de réduire le coût d'un supercalculateur de plusieurs millions d'euros à celui de quelques centaines, ce qui rend ce genre de machine accessible aux écoles et aux universités.

On a même vu apparaître un kit complet constitué de quatre Raspberry Pi Zéro reliés à un Raspberry Pi de taille standard : le Cluster HAT de la société Pimoroni. Il permet de faire des premiers essais d'exécution de programmes sur plusieurs machines en parallèle.

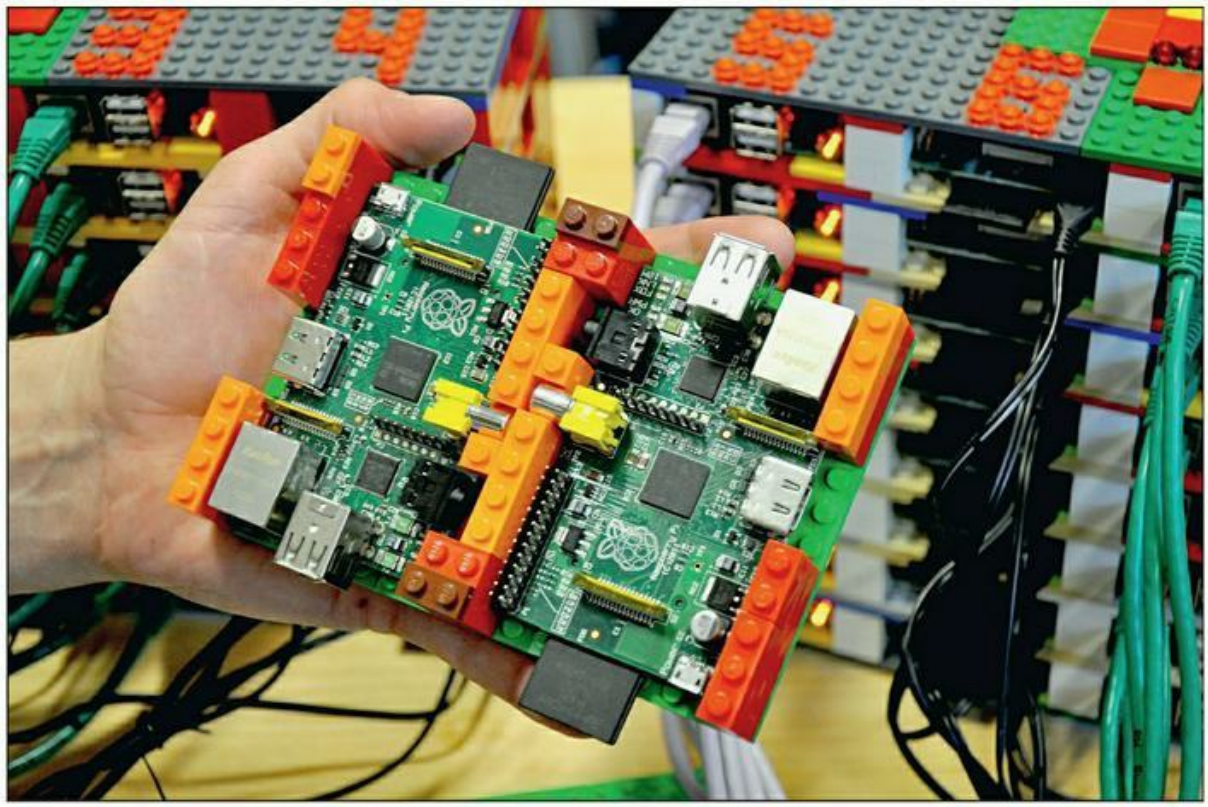


Figure 1.1 : Deux circuits Raspberry Pi du supercalculateur de l'université de Southampton (les autres étant visibles dans le fond).

Le RasPi sert aussi à créer des stations météo, des appareils de fitness, des consoles de jeu vidéo, des lecteurs d'audiolivres, des skateboards électriques, et bien d'autres choses.

Ces projets font la une des journaux, mais une autre histoire, bien que moins visible, est encore plus importante : elle concerne ces milliers de personnes de tous âges qui découvrent ou redécouvrent la science informatique avec un circuit Raspberry Pi.

Les deux auteurs de ce livre utilisaient déjà des ordinateurs dans les années 1980, à l'époque des premiers ordinateurs personnels. Les machines étaient beaucoup moins ergonomiques qu'aujourd'hui. Quand

vous les mettiez sous tension, vous faisiez face à un curseur clignotant. Il fallait savoir quoi saisir pour obtenir un premier résultat. Toute une génération a ainsi grandi avec un minimum de connaissances du contrôle d'un ordinateur et de la création d'un programme. Entretemps, les ordinateurs sont devenus beaucoup plus faciles à utiliser, avec l'arrivée de la souris et de la fenêtre graphique. Nous avons donc perdu ces compétences puisqu'elles n'étaient plus nécessaires.

Le concepteur du Raspberry Pi, Eben Upton, avait remarqué cette baisse du niveau de compétences lorsqu'il travaillait au laboratoire informatique de l'université de Cambridge en 2006. Les candidats étudiants aux cours de sciences informatiques avaient de moins en moins d'expérience en programmation par rapport à leurs aînés. Avec ses collègues, Upton a alors eu l'idée de créer un ordinateur qui serait équipé de tous les outils nécessaires pour programmer et se vendrait pour environ 25 euros. La machine devait être capable de faire aussi des choses attrayantes pour que les gens aient envie de s'en servir, tout en étant assez robuste pour survivre à des promenades répétées dans des sacs à dos d'étudiants.

Cette idée a été le point de départ d'une aventure de six ans qui a donné naissance au Raspberry Pi tel que vous pouvez sans doute le voir sur votre bureau (puisque vous avez acheté ce livre). Le circuit a été lancé en février 2012 et s'est vendu à plus d'un demi-million d'exemplaires à la fin du trimestre. Début 2018, ce sont plus de 15 millions d'unités qui ont trouvé leur propriétaire, dans les écoles, les entreprises et les foyers, dont 10 millions au Royaume-Uni. C'est l'ordinateur anglais le plus vendu de tous les temps !

Premier contact avec le Raspberry Pi

Quand vous observez votre Raspberry Pi, vous voyez un circuit imprimé d'environ la taille d'une carte de crédit parsemé de composants et de connecteurs (Figure 1.2). Dans une époque où la plupart des équipements informatiques sont devenus lisses et brillants, le circuit du RasPi, piquant et parsemé de petites écritures en blanc, ressemble à un alien. Mais justement, son aspect constitue une part de son pouvoir d'attraction. Preuve en est que la plupart des boîtiers que vous pouvez acheter pour emballer votre Raspberry Pi sont transparents, parce que les gens aiment l'admirer.

Figure 1.2 : Un Raspberry Pi 3 modèle B (en haut à gauche), un modèle A+ (en haut à droite) et un Pi Zéro W (en bas).

Depuis ses débuts, Raspberry Pi a beaucoup évolué, en termes de mémoire et de puissance du processeur. La famille est devenue assez grande pour que l'on ne sache pas avec lequel commencer. Voici une visite guidée qui va vous aider à choisir :

» **Raspberry Pi 3 modèle B** : C'est actuellement le plus récent et le plus puissant de tous les RasPi. D'après la fondation Raspberry Pi, il est dix fois plus rapide que la carte Raspberry Pi originale. Il embarque 1 Go de mémoire, quatre ports USB, le Wi-Fi, le Bluetooth et un port Ethernet. Son connecteur d'entrées-sorties GPIO offre 40 broches (les premiers modèles n'en avaient que 26) pour y brancher vos composants électroniques. La taille n'a pas augmenté, c'est celle d'une carte de crédit. Le stockage de données et le support du système est assuré par une carte mémoire au format MicroSD. En cas de doute, optez pour ce modèle qui permet, pour

environ 35 euros, de disposer d'un véritable nano-ordinateur.

Le suffixe « modèle B », est un hommage rendu aux émissions de microinformatique anglaises de la BBC des années 1980. La machine utilisée à l'époque pour pratiquer était la BBC Micro. Pour dix fois plus cher qu'un Raspberry Pi, vous aviez 8 000 fois moins de mémoire. Que de changements en 35 ans !

» **Raspberry Pi 1 modèle A+** : Cette version allégée est réservée aux projets devant consommer le moins possible, donc alimentés par piles. Bien qu'ancienne, elle est parfaite pour les projets autonomes, par exemple en pleine nature. Pas de prise Ethernet et un seul port USB (mais une multiprise USB est toujours possible). Le connecteur GPIO est complet, avec 40 broches. Embarquant 512 Mo de mémoire vive, elle est vendue environ 20 euros. La carte est un peu moins longue que le standard (6,5 cm).

» **Raspberry Pi Zéro :** La fondation Raspberry Pi a stupéfait tout le monde en choisissant d'offrir cette carte avec chaque exemplaire de son magazine *The MagPi*. Mesurant 6,5 cm sur 3 de large, la Pi Zéro est très légère tout en offrant 512 Mo de mémoire et un port Micro USB. Pour utiliser le connecteur GPIO, il faut souder des broches (nous en parlons dans le Chapitre 16). Vendue 5 dollars, la Raspberry Pi Zéro est très souvent en rupture de stock, car la demande est énorme. Notez que s'il faut aussi acheter un

adaptateur Mini HDMI et un adaptateur Micro USB, le prix de revient double ou triple et vous vous retrouvez avec des fils partout.

- **Raspberry Pi Zéro W** : Lancée en février 2017, la variante Raspberry Pi Zéro W résout les problèmes de fils en incorporant le Wi-Fi et le Bluetooth, ainsi que la gestion du module Raspberry Pi Camera. Vendue environ 10 euros, cette variante est un bon premier achat dans la famille Raspberry Pi, surtout si vous n'avez pas tout de suite besoin de vous brancher au connecteur GPIO, qui n'est pas soudé sur les Zéro et Zéro W. C'est notre deuxième choix d'achat après le Raspberry Pi modèle 3.

Les cartes plus anciennes restent disponibles tant qu'il y a assez de demandes et vous pouvez en acheter d'occasion. Par principe, plus une carte est récente, plus elle est puissante. Les améliorations les plus conséquentes sont l'augmentation de la mémoire vive et le remplacement du microprocesseur. De nombreux domaines d'utilisation des RasPi ne demandent pas tellement de puissance, ce qui permet d'y utiliser un modèle un peu plus ancien. Voici quelques-uns de ces prédécesseurs :

- **Raspberry Pi 1 modèle B 256 Mo :** Cette carte a souvent été appelée modèle B, mais c'était la première à être lancée, en février 2012. Elle offre un port Ethernet, deux ports USB et un lecteur de carte SD grand format.

- **Raspberry Pi 1 modèle B 512 Mo :** Lancée en octobre 2012, elle ne se distingue de la précédente que par un doublement de la mémoire vive, ce qui a profité aux applications de traitement graphique.

- **Raspberry Pi 1 modèle A** : La modèle A a été lancée plus tard, en février 2013, pour proposer

une version allégée, moins chère et moins gourmande en énergie. Un seul port USB, pas de port Ethernet et 256 Mo de mémoire.

- **Raspberry Pi 1 modèle B+** : La B+ de juillet 2014 a été annoncée par la fondation Raspberry Pi comme « l'ultime évolution de la Raspberry Pi des origines ». Elle offre 512 Mo de mémoire, quatre ports USB, un plus grand nombre de broches sur le connecteur GPIO, elle consomme moins et améliore la qualité de la sortie audio par rapport à la modèle B. C'est le premier modèle à adopter le format MicroSD qui est dorénavant le standard Raspberry Pi.
- **Raspberry Pi 2 modèle B** : Proposé en février 2015, ce modèle offrait le double de mémoire par rapport au modèle B+, soit 1 Go. Plus de performances pour un encombrement inchangé. Les performances des Raspberry se sont aussi améliorées grâce à des astuces techniques dans les logiciels et les composants. Dès le premier contact, la Pi 2 offre un vrai avantage de vitesse par rapport au modèle B+.

Si vous choisissez de travailler avec n'importe quelle carte plus ancienne que la modèle B+, vous devez utiliser comme carte mémoire une carte SD au format normal, et non Micro. De plus, vous n'avez accès qu'à 26 broches sur le connecteur GPIO au lieu de 40. Les extensions actuellement commercialisées risquent fort de ne pas être compatibles avec les anciennes cartes. Vérifiez tout cela avant de vous engager.

Dans ce livre, nous donnons quelques conseils par rapport aux anciennes cartes, mais nous supposons que vous disposez au minimum d'un modèle B+. Si possible, procurez-vous la carte la plus récente disponible, par exemple la Raspberry Pi 3.

Certains revendeurs proposent le module appelé Raspberry Pi Compute Module, qui est d'abord destiné aux ingénieurs pour les projets industriels

tels que les systèmes embarqués. Nous ne faisons que citer ce module ici, et nous n'en parlons pas plus dans la suite du livre. Ce n'est certainement pas le genre de matériel avec lequel il faut débuter.

Le concept de nano-ordinateur très bon marché a pu se concrétiser sous la forme du Raspberry Pi, en premier lieu grâce aux avancées technologiques des dernières années dans le domaine du traitement de données pour appareils mobiles. Le microcontrôleur au centre de la carte est un processeur Broadcom BCM2835, 2836 ou 2837. Il combine une unité de traitement (CPU) de type ARM et un processeur graphique vidéo (GPU) Core IV. Les deux processeurs partagent le même espace mémoire. Le processeur graphique GPU est suffisamment puissant pour permettre la lecture de vidéo en qualité Blu-ray.

Au lieu d'adopter un des systèmes Windows ou macOS (coûteux), le Raspberry Pi se sert d'un système d'exploitation nommé Linux. C'est un des meilleurs exemples des capacités du mouvement des logiciels libres et open source. L'approche est radicalement différente de celle des logiciels commerciaux. Au lieu d'être la propriété et la chasse gardée d'une entreprise à but lucratif, et donc avec des codes source secrets, Linux a été créé par des volontaires et des entreprises bénévoles de façon coopérative. Chacun peut étudier le code source du système et le modifier (le code source ressemble à une recette de cuisine). De plus, Linux est totalement gratuit, et vous pouvez le donner autour de vous.

N'espérez pas pouvoir utiliser sur votre Raspberry Pi les logiciels dont vous disposez sur vos autres ordinateurs. Il ne permet pas d'utiliser les logiciels prévus pour Windows ou pour macOS. Sous Linux, un certain nombre de logiciels Linux pourront être utilisés, et ce sont en général des logiciels gratuits.

Possibilités du Raspberry Pi

Le RasPi (nous abrégerons) est un véritable ordinateur qui permet de faire presque tout ce que vous avez l'habitude de faire avec un ordinateur de bureau.

Dès que vous le mettez sous tension, vous accédez à un environnement graphique avec des fenêtres et la souris (Chapitre 4), mais aussi à une interface en mode texte de type Terminal (Chapitre 5). Vous pouvez vous en servir pour naviguer sur Internet (Chapitre 4), faire du traitement de texte et des calculs (Chapitre 6) ou retoucher des photos (Chapitre 7). Vous pouvez le transformer en centre multimédia pour lire de la musique ou des vidéos (Chapitre 8) ou jouer à des jeux vidéo. C'est l'outil idéal pour les loisirs à la maison, mais c'est également un ordinateur utilisable

de manière professionnelle pour écrire son courrier, gérer ses comptes bancaires et payer ses factures en ligne. Vous pouvez même le transformer en instruments de création musicale en temps réel avec Sonic Pi (Chapitre 14.)

Mais le domaine de prédilection du Raspberry Pi est encore différent : la découverte du fonctionnement des ordinateurs et la création de montages électroniques. Il est livré avec le langage d'initiation Scratch (Chapitre 9) qui permet à tous de fabriquer ses propres jeux et animations, tout en apprenant certains des concepts fondamentaux de la programmation d'un ordinateur.

Vous disposez également du langage Python (Chapitre 11) qui est un langage professionnel utilisé par YouTube, Google et même l'entreprise qui a créé les effets spéciaux des films *Star Wars* (Industrial Light & Magic), et beaucoup d'autres.

Le circuit est doté d'un connecteur d'entrées/sorties à usage général nommé GPIO. C'est ce connecteur qui permet de brancher vos montages électroniques au Raspberry Pi, afin que ce dernier pilote les périphériques et lise et écrive des signaux électroniques. Nous verrons dans la Partie V comment construire plusieurs jeux et montages électroniques que vous contrôlerez avec le Raspberry Pi.

Se procurer un Raspberry Pi

En résultat de son énorme succès, le Raspberry Pi peut dorénavant être acheté chez de nombreux revendeurs. En plus des magasins installés en France, vous pouvez vous tourner vers les revendeurs internationaux tels que Pimoroni (`www.pimoroni.com`), Pi Hut (`https://thepihut.com`) ou Adafruit (`www.adafruit.com`). Vous trouverez également votre bonheur chez les grands distributeurs d'électronique que sont RS Components (`www.rs-components.com`) et Element14 (`www.element14.com`).

Souvent, les revendeurs proposent des kits regroupant une carte Raspberry Pi avec un certain nombre d'accessoires et de composants. Dans certains cas, cette solution est la plus pratique, mais ce ne sera certainement pas la moins coûteuse pour débuter.

Accessoires indispensables

Les concepteurs du Raspberry Pi ont vraiment cherché à réduire son coût le plus possible afin de rester dans une fourchette entre 25 et 35 euros. En conséquence, vous devez récupérer quelques périphériques ou en acquérir. En effet, le terme « récupérer » n'est pas anodin, car les plus coûteux des périphériques qu'il vous faut sont ceux dont la plupart des gens disposent déjà en double dans un coin de leur garage ou qu'ils peuvent facilement emprunter sur une autre machine ou à un collègue. Si le Raspberry Pi est destiné à devenir votre deuxième ordinateur, vous disposez déjà de la plupart des périphériques dont il a besoin. Cela dit, certains ne sont pas totalement compatibles avec le RasPi, et vous devrez acheter un appareil compatible.

Voici la liste complète des équipements périphériques dont vous aurez besoin :

- **Écran** : Le Raspberry Pi offre une sortie vidéo haute définition sur un connecteur HDMI. Si votre écran possède une entrée HDMI, vous le connectez directement au RasPi. Dans le cas contraire, il dispose peut-être d'une entrée DVI et il suffira d'acheter un convertisseur HDMI vers DVI sur lequel vous brancherez le câble HDMI. Les anciens écrans VGA requièrent un appareil pour convertir le signal HDMI vers le VGA. Si vous comptez en acheter un, vérifiez d'abord qu'il est compatible. Il ne s'agit ici pas simplement d'un câble, mais d'un circuit actif qui va convertir le signal HDMI vers le format VGA. Si votre écran utilise un connecteur bleu avec 3 rangées de broches, c'est sans doute un connecteur VGA. Ce genre de convertisseur n'est généralement pas donné. C'est pourquoi Geert van Loo a créé un appareil qui permet de connecter un moniteur VGA grâce à quelques broches du connecteur GPIO du Raspberry Pi. Il a rendu publics ces schémas, ce qui permet à n'importe qui de les utiliser pour fabriquer et commercialiser des

adaptateurs. Pour en savoir plus, voyez la page https://www.raspberrypi.org/blog/gert-vga-adapter/. Vous savez que votre moniteur est de type VGA s'il utilise un connecteur bleu avec trois rangées de cinq broches.

» **Téléviseur** : Vous pouvez connecter votre Raspberry Pi à un téléviseur Haute Définition *via* son entrée HDMI. L'image devrait être tout à fait correcte. Si vous voulez réutiliser un vieux téléviseur, vous pouvez éventuellement vous en servir, car le RasPi possède une sortie vidéo composite sur le connecteur RCA. Nous avons essayé, et l'image apparaît, même si sa qualité est assez faible, et parfois illisible. Si le recyclage d'un téléviseur est votre seule solution, consultez l'Annexe A qui donne quelques astuces pour améliorer la configuration et donc la lisibilité. Choisissez de préférence un écran informatique. Il vous faut le câble approprié à votre modèle de Raspberry : les modèles A et B possèdent une prise vidéo RCA, mais tous les modèles ultérieurs proposent la sortie vidéo RCA sur la prise casque.

» **Concentrateur USB ou hub** : Le RasPi possède un, deux ou quatre ports USB selon le modèle. L'essentiel est d'y brancher un concentrateur ou hub USB autoalimenté, et cela pour deux raisons. Tout d'abord, vous allez avoir besoin de connecter d'autres périphériques en plus du couple clavier/souris qui monopolise déjà deux prises. Ensuite, le concentrateur autoalimenté va fournir le courant pour tous les périphériques que vous branchez, ce qui évite de surcharger l'alimentation

du Raspberry Pi. Vérifiez donc que le concentrateur hub que vous achetez possède son alimentation indépendante de celui du Raspberry Pi.

- **Clavier et souris USB** : Seuls les claviers et souris USB sont reconnus. Si vous disposez encore d'un périphérique au format PS/2 (connecteur rond), vous pouvez l'oublier.

Si vous constatez que le RasPi fonctionne de façon instable, la cause en est souvent un clavier qui consomme trop de courant. Évitez les claviers avec tout ce rétroéclairage de touches, très gourmands en énergie.

- **Carte mémoire SD** : Il n'y a pas de disque dur dans le Raspberry Pi, et c'est pourquoi son disque système est en fait une carte mémoire SD. Vous en avez déjà utilisé probablement pour votre APN, mais il faut prévoir une capacité minimale de 4 Go. De nos jours, le prix des cartes mémoire a tellement baissé que vous pouvez directement opter pour une 8 Go ou même une 16 Go. Si vous comptez utiliser votre Raspberry comme centre multimédia avec le système LibreELEC, une carte de 4 Go pourra suffire (voyez le Chapitre 8 à ce sujet). Notez que comparativement au disque dur d'un ordinateur, ce n'est pas encore beaucoup d'espace, mais vous pourrez connecter facilement un disque dur externe à votre Raspberry Pi. Sachez que les cartes mémoire SD et MicroSD existent en différentes classes de qualité et de vitesse d'accès. La fondation Raspberry Pi conseille d'utiliser au minimum une carte en classe 6, mais certains revendeurs proposent directement la classe 10. Pour gagner du temps,

vous pouvez même acheter une carte sur laquelle le système de démarrage NOOBS est déjà installé (nous en parlerons dans le chapitre suivant). Dans ce cas, c'est en général une carte de 8 ou de 16 Go au format microSD qui est proposée. Les cartes sont toujours fournies avec un adaptateur, ce qui permet de les insérer dans un lecteur de carte à l'ancien format. (Figure 1.3).

» **Lecteur de carte SD pour le PC** : La plupart des PC et Mac disposent de nos jours d'un lecteur de carte SD interne, qui vous permet notamment de transférer les photos de votre APN. Si ce n'est pas le cas de votre machine, envisagez d'acheter un lecteur de carte SD. Il va vous servir à implanter l'image du système Linux sur une carte SD qui permettra de démarrer le Raspberry Pi. En revanche, vous ne pourrez pas copier les fichiers de votre Raspberry vers Windows. Vous pouvez aussi acheter une carte SD contenant le système Linux préinstallé comme nous l'avons dit plus haut. Cela vous évite d'acheter le lecteur de carte, mais vous ne pourrez pas faire des essais en installant d'autres systèmes sur la carte SD (voyez le Chapitre 2). Comme déjà dit, vous pouvez acheter une carte microSD et son adaptateur sur laquelle tous les logiciels pour le Raspberry Pi sont déjà en place. Mais vous pouvez aussi créer une copie de vos fichiers Raspberry Pi avec un lecteur de carte, comme décrit dans le Chapitre 4.

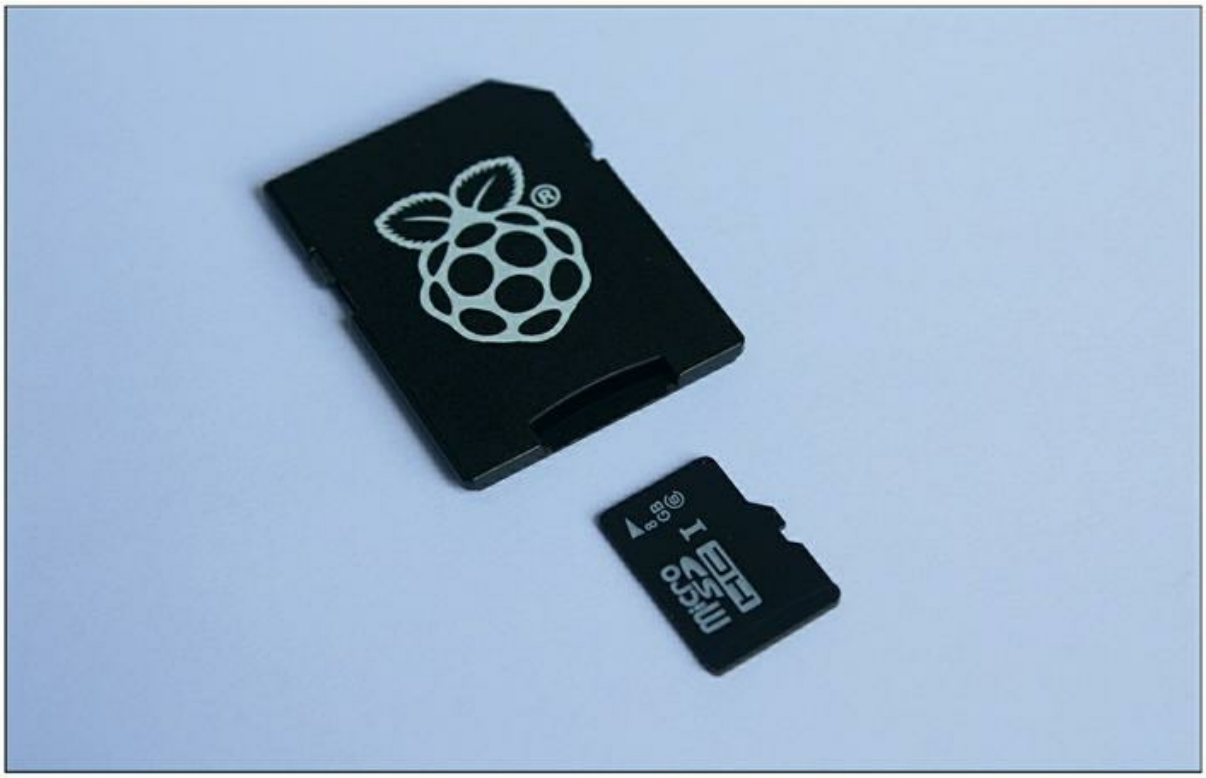


Figure 1.3 : Une carte Micro SD avec son adaptateur.

- **Clé USB** : Les clés mémoire USB sont devenues très bon marché. Les modèles 64 Go sont devenus envisageables. Une telle clé constitue donc un complément idéal pour votre Raspberry Pi. Elle vous permettra de transférer des fichiers entre votre PC ou votre Mac et votre Raspberry Pi.
- **Adaptateur USB Wi-Fi** : Les anciens modèles A et A+ n'ont pas de prise Ethernet. Il vous faut donc obligatoirement un adaptateur USB Wi-Fi. Vous en avez peut-être déjà un pour votre ordinateur portable. Certains ne sont pas compatibles avec le Raspberry, mais les revendeurs qui proposent les adaptateurs Wi-Fi vérifient la compatibilité. Il existe même un adaptateur Wi-Fi officiel que proposent certains revendeurs.
- **Disque dur externe** : S'il vous faut beaucoup de place, par exemple pour profiter de vos fichiers

audio et vidéo sur votre Raspberry, il suffit de connecter un disque dur externe à un port USB. Ce disque doit être autoalimenté ou bien le concentrateur hub USB doit être suffisamment efficace pour correctement alimenter la carte.

- **Module caméra Raspberry Pi :** Le succès du Raspberry Pi a poussé des créateurs à proposer toutes sortes d'extensions. Le module Camera est l'une des rares extensions à avoir été conçue et produite par la fondation Raspberry Pi elle-même. Il s'agit d'une caméra à mise au point fixe de 8 mégapixels qui permet de filmer en HD et de prendre des photos. Il existe même une variante dotée d'un filtre infrarouge (PiNoIR) qui permet de filmer la vie sauvage en pleine nuit ou d'ajouter de drôles d'effets spéciaux en plein jour.
- **Haut-parleurs** : Le Raspberry Pi offre une sortie audio standard stéréo au format jack 3,5 mm qui convient à la plupart des casques et haut-parleurs pour PC. Nous donnons des détails pour les connexions à une chaîne stéréo dans le Chapitre 3. Notez que vous n'avez pas besoin de casque ni de haut-parleur si vous utilisez la connexion HDMI, car l'audio est transmis en même temps que la vidéo. En revanche, cela ne fonctionne pas si votre écran est DVI, car le son n'est pas transmis en DVI.
- **Alimentation** : Le Raspberry Pi est alimenté par un connecteur microUSB, ce qui le rend en théorie compatible avec la plupart des chargeurs de téléphones portables. En pratique, la plupart de

ces appareils ne peuvent pas fournir un courant suffisant (il faut prévoir jusqu'à 700 mA pour les anciens modèles et même 2500 mA pour le modèle 3B), et cela peut rendre le fonctionnement du Raspberry Pi instable. Même la résistance du câble d'alimentation joue un certain rôle et peut empêcher les périphériques tels que la souris de bien fonctionner. Vérifiez le courant qu'offre le chargeur que vous prévoyez d'utiliser. Il existe une alimentation électrique Raspberry Pi officiel. Pour un projet qui doit être autonome, vous pouvez alimenter votre carte avec un jeu de batteries, tels que celles qui servent à recharger des smartphones. Vous n'aurez aucune surprise en achetant le chargeur en même temps que votre circuit Raspberry Pi. N'essayez pas d'alimenter le RasPi en branchant son connecteur micro-USB à un port USB du PC, car l'ordinateur fournit généralement un courant insuffisant sur ses ports USB. Vous pouvez enfin alimenter votre RasPi par les broches GPIO, mais vous pourriez endommager le circuit si vous vous trompez dans la tension ou le courant. Si vous envisagez cette solution, nous vous conseillons d'utiliser une carte d'extension HAT qui viendra s'enficher sur le connecteur GPIO pour alimenter de manière fiable le circuit. La fondation Raspberry Pi conseille de n'utiliser des piles ou batteries pour l'alimentation que si vous vous sentez à l'aise, car il y a des risques d'endommager le circuit. Pour tout détail à ce sujet, consultez la page technique à l'adresse `www.raspberrypi.org/help/faqs/`.

N. d. T. : Lorsque le hub USB est de bonne qualité, vous pouvez alimenter le RasPi depuis le hub, mais n'ajoutez pas trop de périphériques dans ce cas.

- **Boîtier** : Vous pouvez tout à fait utiliser votre circuit tel qu'il est, mais de nombreuses personnes préfèrent le protéger des éclaboussures et des éventuels courts-circuits que peuvent causer des trombones égarés sur le bureau. Vous trouverez sur les sites de vente aux enchères des boîtiers transparents pour la plupart, ce qui permet d'admirer le circuit et de voir les LED. Ces boîtiers sont généralement livrés en kit. L'un des plus esthétiques est le Pibow Coupe de chez Pimoroni (www.pibow.com) avec ses couches de plastique à effet arc-en-ciel (voir Figure 1.4). Il a été conçu par la personne qui a inventé le logo du Raspberry Pi, Paul Beech. Vous pouvez soit refermer le tout de façon hermétique, laisser une lumière pour la caméra ou garder l'accès aux broches du connecteur GPIO. Cela dit, l'achat d'un boîtier n'est pas obligatoire. Vous pouvez aussi créer votre propre boîtier avec du carton ou des briques Lego. Dans tous les cas, vérifiez que vous pouvez toujours accéder au connecteur GPIO afin de pouvoir brancher vos montages électroniques, comme nous vous le montrons dans la Partie V du livre.

Figure 1.4 : Le boîtier Pibow Coupe pour Raspberry Pi 2, Pi 3 ou modèle B+.

» **Câbles** : Il vous faut au minimum 4 câbles : un câble HDMI (pour un écran HDMI ou DVI, avec un adaptateur HDMI vers DVI dans le dernier cas) ou un câble vidéo RCA pour un vieux téléviseur, un câble audio si vous ne vous êtes pas connecté en HDMI, un câble Ethernet (si nécessaire) et un câble Micro USB pour l'alimentation. Les derniers modèles Raspberry Pi mettent à disposition le signal vidéo RCA sur une prise jack de type casque (3,5 mm), alors que les anciens modèles offraient une prise RCA dédiée. Le câble sera bien sûr différent selon la carte dont vous disposez. Si vous avez un Raspberry Pi zéro, il faut également un convertisseur pour la prise Mini HDMI et un autre pour la prise Micro USB. Vous trouverez tous ces câbles sur le site sur lequel vous achetez votre Raspberry Pi ou chez n'importe quel revendeur de composants. Les autres câbles éventuels (par exemple pour relier la sortie audio à des haut-

> parleurs ou pour le concentrateur USB) sont normalement livrés avec ces appareils.

Le Raspberry Pi a été conçu pour être utilisé avec la plupart des accessoires dont vous disposez déjà, afin de réduire les coûts de l'ensemble. En pratique, tous les appareils ne sont pas compatibles, notamment certains concentrateurs hubs USB, certains claviers et souris, et les problèmes qu'ils engendrent peuvent être parfois difficiles à détecter. Il existe des concentrateurs ou multiprises USB qui renvoient du courant vers le Raspberry Pi par le port USB, ce qui peut gravement endommager votre carte si le courant renvoyé est trop important.

N'hésitez pas à consulter la liste des périphériques compatibles et non compatibles à l'adresse suivante : http://elinux.org/RPi_VerifiedPeripherals. Consultez aussi les avis sur Internet pour savoir si d'autres personnes ont eu des problèmes pour utiliser un appareil en particulier avec le Raspberry Pi.

Dans tous les cas, vous prenez moins de risques en achetant vos accessoires auprès du revendeur chez lequel vous achetez votre Raspberry Pi.

Au cas où, prévoyez un petit budget pour vos accessoires. Le Raspberry Pi est très bon marché, mais si vous êtes obligé d'acheter un clavier, une souris, un concentrateur USB alimenté, une carte mémoire SD et tous les câbles, cela peut doubler ou triple le budget total. Voilà pourquoi il vous faudra éventuellement chercher parmi ceux dont vous disposez déjà, quitte à constater que tout n'est pas compatible.

Chapitre 2

Télécharger le système d'exploitation

DANS CE CHAPITRE :

- **Présentation de Linux**
- **Choix d'une distribution Linux**
- **Téléchargement et décompression de NOOBS**
- **Copie de NOOBS sur carte SD**
- **Implantation d'un système sur une carte mémoire SD par flashage**

Avant de pouvoir faire quoi que ce soit avec votre Raspberry Pi, vous devez pouvoir le faire démarrer avec une carte mémoire système. Le système d'exploitation est le logiciel qui va animer l'ordinateur et en contrôler le fonctionnement. C'est dans le système que vous utilisez des applications pour par exemple gérer vos fichiers et réaliser des activités comme rédiger un courrier ou naviguer sur le Web. Ces applications s'appuient sur le système pour tous leurs besoins matériels. Cette approche n'est pas spécifique au Raspberry Pi. Votre ordinateur portable ou de bureau fonctionne avec le système Microsoft Windows ou macOS. Votre smartphone ou votre tablette fonctionne avec le système iOS, Android ou autre.

Nous allons découvrir dans ce chapitre le système d'exploitation Linux, qui est le plus utilisé sur Raspberry Pi. Nous allons voir comment préparer une carte mémoire microSD ou SD permettant de faire démarrer le système. La préparation devra se faire sur un autre ordinateur, sous Windows, sous macOS ou sous Linux.

L'essentiel est d'avoir accès à un lecteur de carte microSD et à une connexion à Internet.

Découverte de Linux

Le système d'exploitation de prédilection pour le Raspberry Pi porte le nom GNU/ Linux, mais vous entendrez plus souvent parle de « Linux ». Ce sera pour certains le premier ordinateur qu'ils vont utiliser sous Linux, mais ce système peut déjà se targuer d'une longue histoire.

En 1984, un certain Américain, Richard Stallman, lance le projet GNU en vue de créer un système d'exploitation et des logiciels que tout le monde pourrait utiliser, copier, donner autour de soi et modifier. C'est ainsi qu'a été lancé le mouvement des logiciels libres (*free software*), presque toujours gratuits (mais ce n'est pas obligatoire). Des milliers de passionnés sont venus participer au projet GNU pour créer une foule de logiciels : outils, applications, et même des jeux.

Richard Stallman avait choisi de concevoir son système d'exploitation pour qu'il reste compatible avec le système Unix, un système d'exploitation pour grands ordinateurs créé dans les années 1970 dans les laboratoires ATT Bell. Grâce à ce choix stratégique, tous les utilisateurs habitués à Unix pouvaient facilement adopter la philosophie des logiciels libres et open source que leur proposait Stallman par son projet GNU.

En 1991 survient le Finlandais Linus Torvalds. Gêné de devoir payer pour découvrir un petit système Unix à but pédagogique (Minix), il décide d'écrire la partie essentielle de son propre système de style Unix, c'est-à-dire le noyau (*kernel*). Ce noyau est l'arbitre général qui fait le lien entre le matériel (capacité de calcul du processeur et espace mémoire) et les applications des utilisateurs. Linus reste à ce jour le décideur ultime quand une personne vient proposer une amélioration du noyau standard Linux, comme l'indique le site de la Fondation Linux qui assure sa promotion et soutient son développement. On y apprend aussi que plus de 12 000 personnes de 1 000 sociétés ont contribué bénévolement à l'amélioration du noyau depuis 2005.

N. d. T. : Vous aurez deviné que le nom « Linux » combine le prénom de l'auteur, Linus, et le nom Unix.

GNU/Linux réunit le noyau Linux et les outils GNU pour former un système d'exploitation complet, fruit des efforts de milliers de personnes sur les deux projets, GNU et Linux. La possibilité pour un tel nombre de gens de travailler ensemble pour créer quelque chose d'aussi complexe qu'un système d'exploitation, puis de le donner à la communauté mondiale, est quasiment un miracle des temps modernes.

Puisque le système peut être modifié et rediffusé sans limites, de nombreuses versions de GNU/Linux sont apparues. Ce sont des distributions, certaines dédiées aux experts, d'autres aux débutants, d'autres aux musiciens, aux graphistes, aux enfants, *etc*. Toutes ne sont pas utilisables sur le Raspberry Pi. Cela dit, les logiciels (applications) sont généralement utilisables dans n'importe quelle distribution Linux. En revanche, les programmes prévus pour Microsoft Windows ou pour macOS ne fonctionnent pas (ou réclament un émulateur qui consomme beaucoup de puissance).

Au sens strict, Linux ne désigne que le noyau du système d'exploitation, mais nous reprenons dans ce livre l'habitude de désigner sous le nom Linux l'ensemble noyau et outils, donc GNU/Linux.

Pour le Raspberry Pi, la distribution Linux à privilégier se nomme Raspbian. Nous la présentons dans le prochain chapitre.

Préparation d'une carte NOOBS

La solution la plus confortable pour faire ses premiers pas avec un Raspberry Pi consiste à utiliser NOOBS. Cet acronyme de *New Out Of the Box* (« tout frais sorti de la boîte ») est également un clin d'œil aux débutants en informatique, qu'on appelle *noobs* en anglais. Ne vous fiez cependant pas à ce nom pour mésestimer les possibilités de NOOBS. Ce système est très facile à implanter sur une carte mémoire, depuis laquelle il permet par un menu d'installer plusieurs systèmes, éventuellement en même temps. Hormis cela, vous trouverez plusieurs versions de Linux et deux versions du centre multimédia Kodi. Je décris les différentes possibilités en détail dans le Chapitre 3.

Comme déjà dit dans le Chapitre 1, il est possible d'acheter une carte mémoire sur laquelle NOOBS est déjà en place, ce qui simplifie les choses. Si c'est votre cas, vous pouvez directement passer au Chapitre 3. Cela dit, il n'est pas inutile de parcourir les pages suivantes pendant l'installation du système. Il est toujours utile de savoir comment créer une carte NOOBS. En un quart d'heure, vous disposez d'une nouvelle carte, ce qui vous évite d'attendre d'être livré par courrier.

Le site officiel de Raspberry Pi comporte des liens permettant de télécharger directement les images de plusieurs systèmes d'exploitation. Il ne reste ensuite qu'à implanter l'image désirée sur une carte mémoire, mais ce genre de fichier image ne peut pas être directement copié sur la carte mémoire. Un travail de reformatage est nécessaire, ce qui correspond à l'opération de flash de la carte. Nous expliquons comment procéder tout à la fin de ce chapitre.

Télécharger NOOBS

Avec le navigateur de votre ordinateur de bureau, rendez-vous à l'adresse suivante :

Deux versions sont proposées. La première contient le système Raspbian qui est le système officiel. Vous n'aurez donc pas besoin de configurer une connexion réseau pour commencer à utiliser votre Raspberry Pi. NOOBS comporte un menu qui permet de sélectionner un ou plusieurs systèmes d'exploitation à télécharger en plus de Raspbian. Ce n'est possible que si vous avez une connexion réseau. Nous conseillons de télécharger NOOBS dans sa version complète pour installer le système Raspbian.

L'autre variante, NOOBS Light, est plus compacte, mais elle réclame une connexion Internet pour pouvoir télécharger les systèmes proposés. C'est la solution à adopter si vous ne comptez pas utiliser le système Raspbian. Bien sûr, sans connexion Internet, cette version allégée est inutilisable.

Formater la carte mémoire SD

Même si la carte est neuve, il est conseillé de procéder à son formatage. Vous pouvez utiliser un programme gratuit fourni par l'association des fabricants de cartes SD. Ce programme existe pour macOS et pour Windows.

Récupérez l'outil à l'adresse suivante :

Vous devez bien sûr accepter les conditions d'utilisation pour pouvoir télécharger.

Si vous utilisez Linux, le formatage de la carte va être réalisé avec l'outil GParted, comme nous l'expliquons plus loin.

Les premiers modèles de Raspberry utilisaient un lecteur de carte au format standard, mais tous les suivants utilisent le format microSD. Les cartes microSD sont presque toujours livrées avec un adaptateur plastique pour les utiliser sur un lecteur au grand format (Figure 1.3).

Vous devez être extrêmement attentif avant de lancer le formatage. En effet, ce que contient la carte mémoire va être totalement supprimé. Pensez à sauvegarder ce contenu si c'est une carte que vous réutilisez. Profitez-en pour vérifier que vous avez bien réalisé une sauvegarde récente de votre disque dur et débranchez tous les disques amovibles inutiles, afin de limiter les risques d'effacement accidentel de données.

Formatage sous Windows

Une fois téléchargé, l'outil de formatage Windows SD Card Formatter se présente sous forme d'un fichier d'installation *.exe*. Il suffit de le double-cliquer pour lancer le programme. Cliquez le bouton **Next**. Le programme propose un lieu d'implantation du logiciel. Acceptez cette proposition et cliquez **Install**. Il est possible que Windows affiche un message confirmant qu'un programme tente de modifier l'ordinateur et vous demande de confirmer que vous acceptez.

Une fois l'outil installé, il n'y a pas d'icône pour le lancer. Servez-vous de la fonction de recherche de Windows. Sous Windows 10, cliquez dans la boîte de la barre des tâches dans le coin inférieur gauche de l'écran pour faire votre recherche. Sous Windows 8, amenez le pointeur de souris dans l'angle supérieur droit pour cliquer dans la loupe. Dans les deux cas, saisissez **SD**. Quand le nom complet du programme apparaît, cliquez-le. (Sur les versions plus anciennes de Windows, vous pouvez lancer **SD Card Formatter** depuis le menu Démarrer. Il est possible qu'il faille confirmer que vous acceptez le lancement du programme.)

La boîte de dialogue de l'outil est montrée dans la Figure 2.1. Ouvrez la liste **Select card** pour choisir l'unité dans laquelle est insérée votre carte mémoire SD. Si rien n'est affiché, utilisez le bouton **Refresh**. Relisez bien le nom de ce que vous avez sélectionné, car l'outil va détruire tout le contenu de cette unité. C'est pour cette raison que nous avons conseillé de déconnecter tous les autres disques et clés USB, afin de limiter les choix et donc les risques.

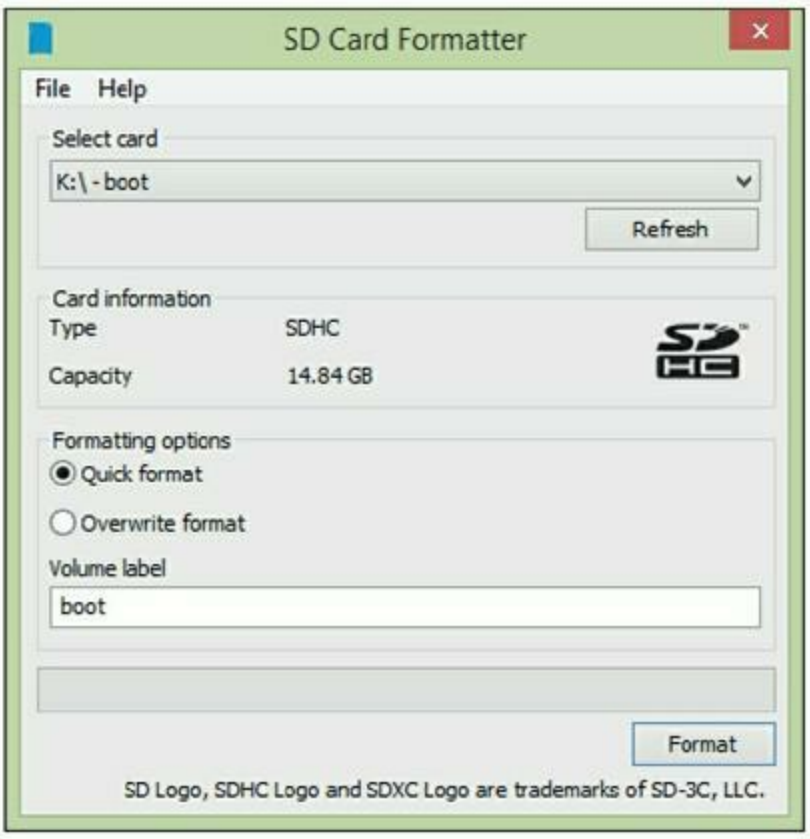


Figure 2.1 : L'outil SD Card Formatter sous Windows.

L'option de formatage **Quick format** accélère le formatage. En revanche, elle ne supprime pas les données éventuelles. Pour autant, n'espérez pas pouvoir les récupérer en cas de fausse manœuvre. Après avoir tout revérifié, cliquez le bouton **Format**. Lors de nos essais, il n'a fallu que quelques secondes.

Formatage sous macOS

La version macOS de l'outil de formatage se présente sous forme d'un paquetage d'installation qu'il suffit de double-cliquer pour installer le produit. Par défaut, le logiciel est installé dans le dossier Applications. Vous devez connaître le mot de passe pour installer le programme, et à chaque fois que vous voudrez l'utiliser.

Insérez la carte mémoire dans le lecteur externe ou interne puis double-cliquez l'icône SD Card Formatter dans le dossier des applications. La boîte de dialogue principale est celle de la Figure 2.2.

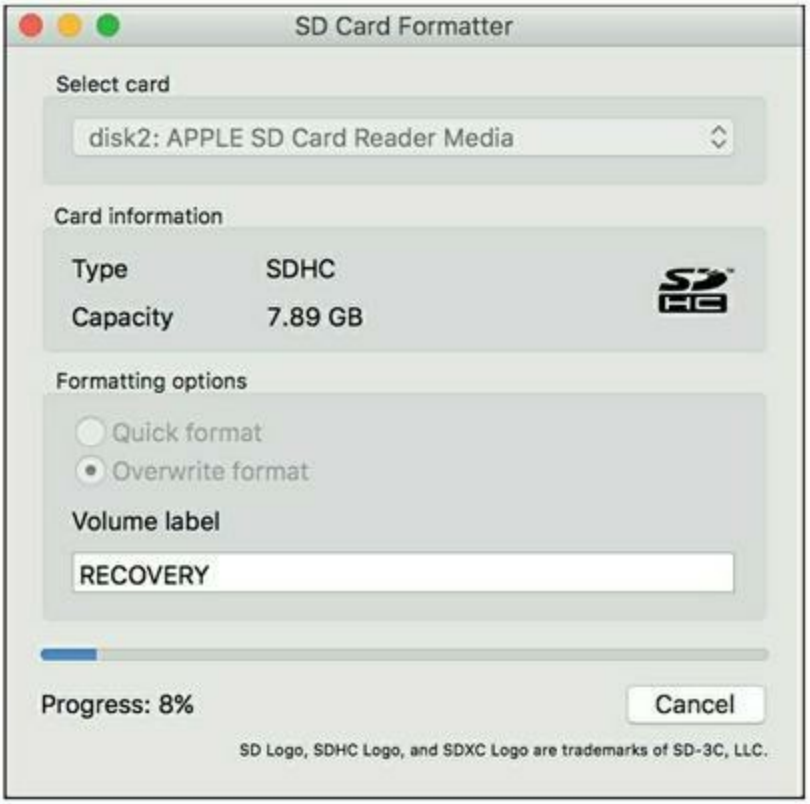


Figure 2.2 : L'outil SD Card Formatter pour macOS.

Regardez bien le contenu de la liste déroulante du haut, **Select card**. Elle doit indiquer une mention suffisant à confirmer que ce qui est sélectionné est bien la carte mémoire SD ou microSD. Si ce n'est pas cette carte qui est affichée, ouvrez la liste pour la sélectionner.

Sélectionnez maintenant l'option **Overwrite Format** puis cliquez le bouton **Format**. Il est inutile de saisir un nom dans la zone **Volume label**, car ce nom va être renseigné automatiquement.

Vous pouvez maintenant aller boire un verre, parce qu'il est possible qu'il faille quasiment une demi-heure pour réaliser le formatage. Cela dit, vous pouvez continuer à utiliser votre machine pendant le traitement, bien que cela le ralentisse encore un peu. Une fois le travail réalisé, vous allez voir apparaître l'icône de la carte mémoire sur le bureau. Notez que si la carte a déjà servi, vous gagnerez du temps en activant l'option **Quick format**.

Formatage sous Linux

La distribution Linux actuellement la plus utilisée se nomme Ubuntu, et c'est celle dont nous nous servons. Les conseils qui suivent sont utilisables avec d'autres distributions, moyennant quelques adaptations.

Pensez à faire une sauvegarde de toutes vos données avant de lancer ce traitement, car il peut par mégarde effacer votre disque dur système.

Voici comment procéder :

1. **Pour préparer la carte mémoire sous Ubuntu, nous allons utiliser l'outil GParted**. S'il n'est pas encore installé, nous allons le récupérer et l'installer au moyen des deux commandes suivantes :

```
sudo apt-get update
sudo apt-get install gparted
```

   Rappelons que ces commandes doivent être émises dans une fenêtre de terminal que vous ouvrez par le menu des applications ou par le raccourci Ctrl + T.

2. **L'outil GParted doit être démarré en tant que superutilisateur root**. Voici la commande à saisir dans le terminal :

   La Figure 2.3 montre l'utilisation de GParted.

3. **Dans le coin supérieur droit de la fenêtre, ouvrez la liste pour sélectionner votre carte SD**.

   Soyez très attentif ici, car vous pouvez en un clin d'œil effacer le contenu de votre disque dur. Dans notre exemple, nous sommes certains que nous avons sélectionné la carte SD parce que la capacité indiquée est de 7,4 Go, alors que notre disque dur principal a une capacité de 500 Go.

4. **L'outil présente les partitions trouvées sur le support mémoire**. Elles doivent toutes être supprimées. Sélectionnez chacune d'elles tour à tour et utilisez le bouton **Delete** de la barre d'outils.

Le bouton **Delete** offre l'aspect d'une interdiction de stationner. Notez que l'opération n'est pas réalisée immédiatement mais stockée dans une file d'attente d'actions. Toutes les actions seront réalisées lorsque vous validerez.

Si vous ne parvenez pas à sélectionner une partition pour la détruire, c'est parce qu'elle possède une icône de clé. Dans ce cas, cliquez droit dans la partition et choisissez **Unmount** dans le menu local.

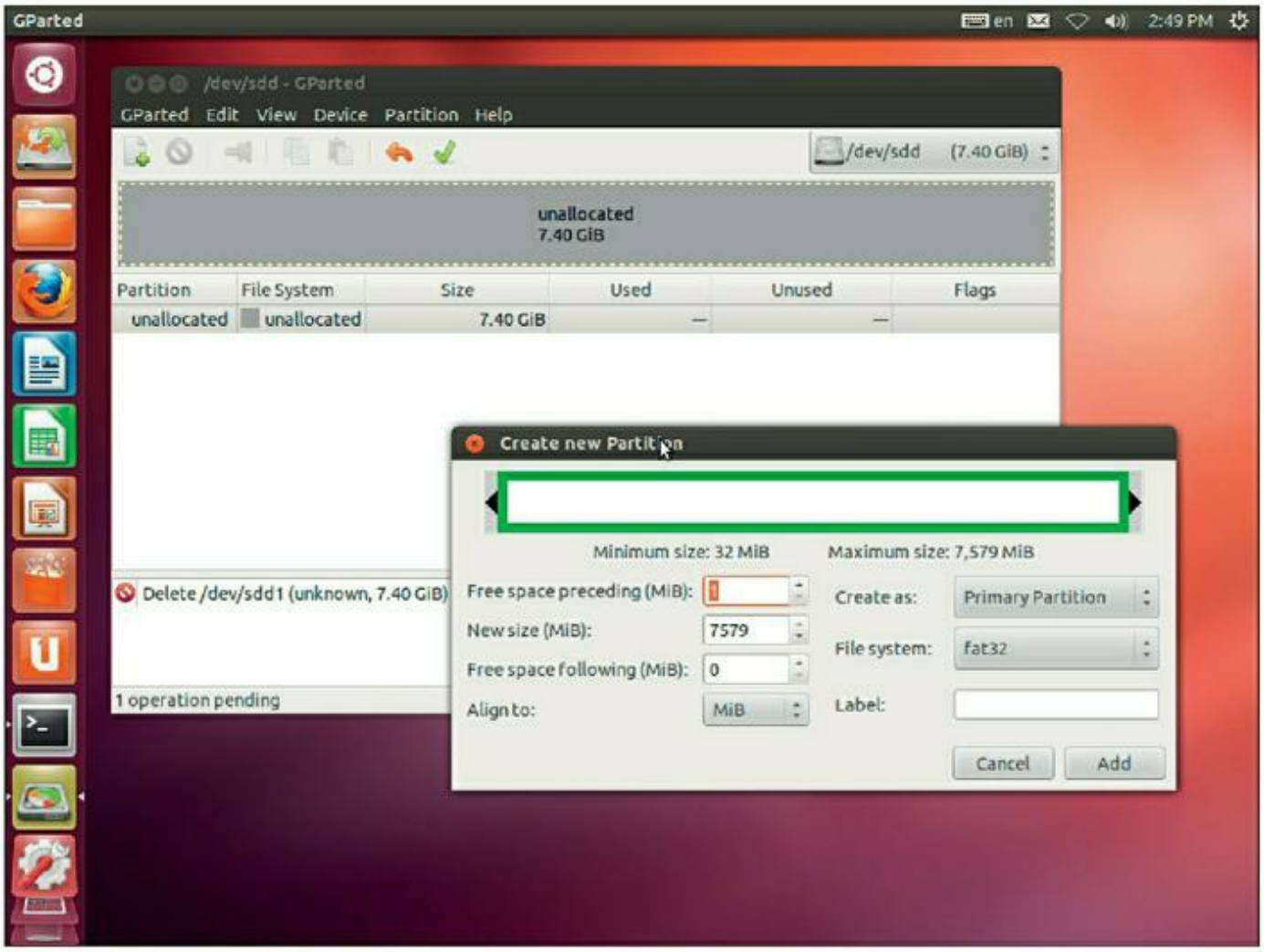


Figure 2.3 : L'outil GParted sur le bureau de Linux Ubuntu.

5. **Vous aboutissez à un seul espace non alloué**. Cliquez le bouton **Add partition**, celui qui montre

une page vide avec un signe +.

Une seconde fenêtre s'ouvre, comme celle visible dans le bas de la Figure 2.3.

6. **Ouvrez le menu pour choisir le système de fichiers FAT32 puis cliquez le bouton Add**.

   Aucune action n'est encore enclenchée.

7. **Dans la barre d'outils, cliquez le bouton contenant une encoche verte, ce qui provoque l'exécution enchaînée de toutes les actions de la file d'attente**. Les partitions sont supprimées puis la nouvelle partition est créée au format FAT32.

Copie de NOOBS vers la carte SD

Vous possédez maintenant une carte SD ou microSD préparée et le fichier compressé .zip de NOOBS que vous avez téléchargé. Il ne reste plus qu'à transférer le contenu de l'archive vers la carte en copiant simplement les fichiers qui se trouvent dans le fichier .zip.

Sous Windows, double-cliquez le fichier .zip de NOOBS puis sélectionnez tous les fichiers qu'il contient et copiez-les vers la carte SD. Pour sélectionner les fichiers, vous disposez du raccourci Ctrl + A ; pour la copie, utilisez Ctrl + C.

Sous macOS, double-cliquez le fichier compressé pour voir apparaître un dossier avec tous les fichiers qu'il contient. Dans le menu **Édition**, choisissez **Sélectionner tout** puis faites glisser tous les fichiers vers l'icône de la carte SD. Patientez ensuite une quinzaine de minutes. Une fois cela terminé, éjectez la carte mémoire SD. La Corbeille a sans doute pris l'aspect d'une icône d'éjection.

Sous Linux, le processus est le même. Vous double-cliquez le fichier compressé .zip de NOOBS, vous sélectionnez tous les fichiers qu'il contient et vous les déposez dans la carte SD.

Si votre Linux n'a pas d'interface graphique, vous pouvez également réussir l'opération depuis la ligne de commande :

1. **Éjectez puis réinsérez la carte pour qu'elle soit automatiquement montée**.
2. **Ouvrez une fenêtre de terminal**.

   Sous Ubuntu, vous pouvez utiliser l'entrée du menu des applications ou le raccourci Ctrl + Alt + T (sous Ubuntu).
3. **Saisissez la commande suivante, en sachant que la dernière lettre est bien un l minuscule :**

   Vous voyez ainsi apparaître tous les disques actuellement détectés (Figure 2.4).
4. **Lisez attentivement cette liste pour repérer quel disque correspond à votre carte SD**.

   Dans notre exemple, deux disques sont présentés. La première ligne de chaque section commence par le mot disque suivi d'une mention */dev/ sdx*. Le premier disque possède une capacité de 500,1 Go, ce qui montre que c'est notre disque dur principal. Le second disque n'offre que 7948 Mo, soit environ 8 Go. C'est bien notre carte SD. Notez le nom de la partition correspondante, dans l'exemple c'est **sdd1**.

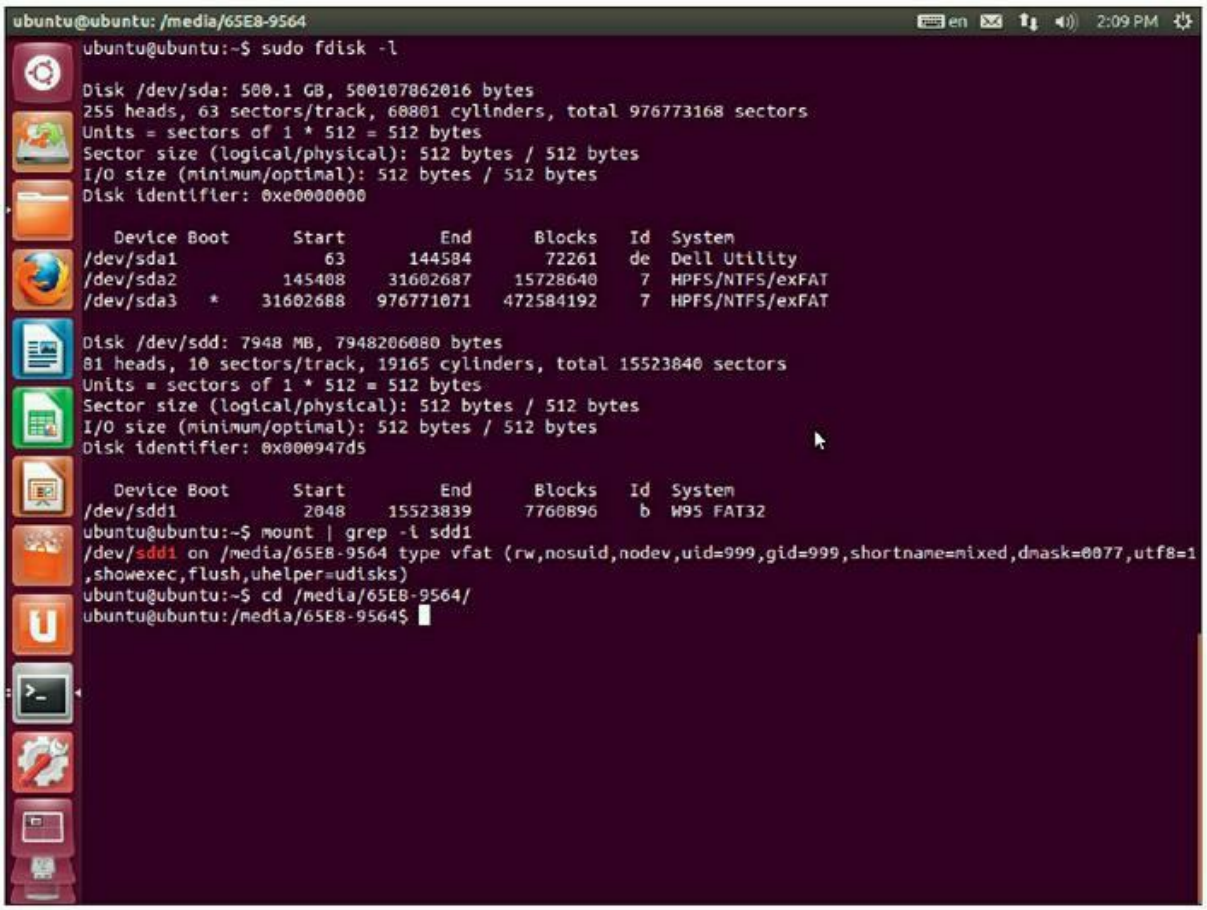

ubuntu@ubuntu: /media/65E8-9564
en 2:09 PM
ubuntu@ubuntu:~$ sudo fdisk -l
Disk /dev/sda: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xe0000000
Device Boot Start End Blocks Id System
/dev/sda1 63 144584 72261 de Dell Utility
/dev/sda2 145408 31602687 15728640 7 HPFS/NTFS/exFAT
/dev/sda3 * 31602688 976771071 472584192 7 HPFS/NTFS/exFAT
Disk /dev/sdd: 7948 MB, 7948206080 bytes
81 heads, 10 sectors/track, 19165 cylinders, total 15523840 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000947d5
Device Boot Start End Blocks Id System
/dev/sdd1 2048 15523839 7760896 b W95 FAT32
ubuntu@ubuntu:~$ mount | grep -i sdd1
/dev/sdd1 on /media/65E8-9564 type vfat (rw,nosuid,nodev,uid=999,gid=999,shortname=mixed,dmask=0077,utf8=1
,showexec,flush,uhelper=udisks)
ubuntu@ubuntu:~$ cd /media/65E8-9564/
ubuntu@ubuntu:/media/65E8-9564$

Figure 2.4 : Affichage de la liste des disques sous Ubuntu.

5. **Cherchez à quel endroit la carte a été montée**.

   La commande **mount** permet de savoir dans quel répertoire a été implantée la racine de la carte mémoire. Dans l'exemple, la partition porte le nom **sdd1**, et nous entrons la commande de la manière suivante :

   Le bas de la Figure 2.4 montre la réponse fournie par l'outil. Ici, la carte a été montée dans le sous-répertoire */media/65E8-9564*.

6. **Servez-vous de la commande cd pour vous placer dans le sous-répertoire de la carte :**

7. **Décompressez directement le fichier compressé NOOBS vers cette carte**.

Suite au téléchargement, le fichier compressé a été stocké dans le dossier */home/ubuntu/Downloads*. La commande suivante permet de les décompresser directement dans la racine de la carte SD :

N'hésitez pas à utiliser le mécanisme basé sur la touche de tabulation qui permet de terminer la saisie dès qu'il n'y a plus d'ambiguïté. Le nom exact du fichier compressé sera sans doute différent de celui de l'exemple quand vous réaliserez cette opération. Comptez environ cinq minutes pour la décompression et la copie.

Utiliser sa carte NOOBS

Une fois que la carte NOOBS est prête, nous pouvons passer à la préparation de notre Raspberry Pi. Dans le prochain chapitre, nous allons d'abord voir comment connecter les périphériques indispensables, comment insérer la carte mémoire et comment terminer l'installation du système.

Créer une carte SD système

Si vous avez créé une carte NOOBS, vous pouvez passer au chapitre suivant. Nous avions cependant dit que certains systèmes d'exploitation n'étaient pas disponibles avec NOOBS, et notamment le système RISC OS. Pour créer une carte de démarrage pour un tel système, il faut télécharger l'image du système d'exploitation puis implanter cette image sur la carte par l'opération de flashage.

Vous trouverez de nombreuses images de systèmes sur le site officiel, dans la section de téléchargements. Vous pouvez même télécharger une image de Raspbian, si vous ne voulez pas utiliser NOOBS.

Pour implanter un système, nous vous proposons d'utiliser l'outil Etcher, qui fonctionne sous Windows, macOS et Linux. Vous le téléchargez à l'adresse suivante :

Une fois que vous l'avez installé, le démarrage montre une seule fenêtre très épurée (Figure 2.5). Voici comment préparer votre carte système :

Figure 2.5 : L'outil Etcher permet de créer une carte SD système.

1. **Du côté gauche de la fenêtre, choisissez le fichier image à implanter**.

   Certains fichiers images sont fournis au format .zip. L'outil Etcher sait les utiliser sans qu'il faille d'abord extraire le contenu.

2. **Dans le panneau central, choisissez l'unité qui contient la carte devant recevoir le système**.

   Ce choix va entraîner la suppression du contenu actuel de ce support de données. Soyez donc vigilant. Déconnectez tous les autres périphériques de stockage pour limiter les risques.

3. **Cliquez enfin le bouton Flash à droite**.

L'image est alors exploitée pour créer tous les fichiers qui vont permettre de démarrer le système avec cette carte SD.

Chapitre 3

Connecter votre Raspberry Pi

DANS CE CHAPITRE :

- **Insérer une carte mémoire SD**
- **Connecter un écran, un clavier et une souris**
- **Connecter le circuit à un routeur ou au Wi-Fi**
- **Connecter une caméra**
- **Configurer le Raspberry Pi avec raspi-config**

Après avoir déballé votre circuit Raspberry Pi, vous serez peut-être désarçonné devant ce petit circuit que vous tenez entre deux doigts. Rassurez-vous : vous allez très facilement le connecter et le mettre en état de marche. Certains auront peut-être besoin de retoucher la configuration (nous en parlons dans l'Annexe A), mais la grande majorité d'entre vous pourront faire fonctionner leur Raspberry Pi dès qu'ils auront connecté tous les périphériques.

Pour bien commencer, vérifiez que vous tenez le circuit du Raspberry Pi comme nous nous y attendons pour pouvoir suivre nos descriptions. Le dessus de la carte correspond au côté sur lequel se trouvent la plupart des composants et des légendes ; c'est le côté montré sur les Figures 3.1, 3.2 et 3.3. Le dessous présente essentiellement des soudures. Orientez votre carte pour que les deux rangées de broches du connecteur GPIO soient horizontales en haut. Dans le cas du Pi Zéro ou du Pi Zéro W, ce sont les emplacements des broches qui doivent se trouver à cet endroit.

La Figure 3.1 décrit les ports et connecteurs pour le modèle B original de la Raspberry Pi. Le modèle A ne s'en distingue que par l'absence de la prise Ethernet en bas à droite.

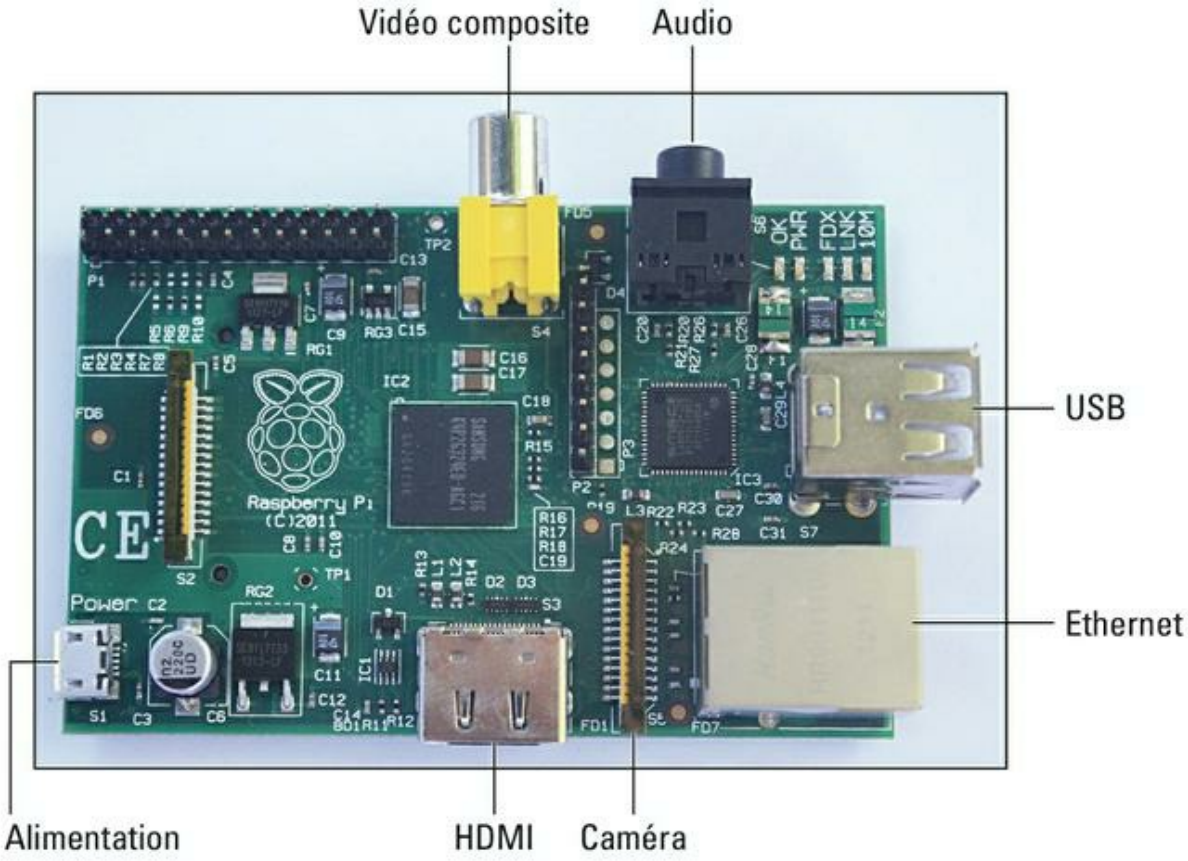


Figure 3.1 : Connexions de la carte originale modèle B (et de la modèle A sans connecteur Ethernet).

La Figure 3.2 montre les connexions pour la Raspberry Pi 3 modèle B, et s'applique également à la Pi 2 et au modèle B+. Le modèle A+ est un peu plus court, mais la configuration est la même, à part l'absence de prise Ethernet et un seul port USB.

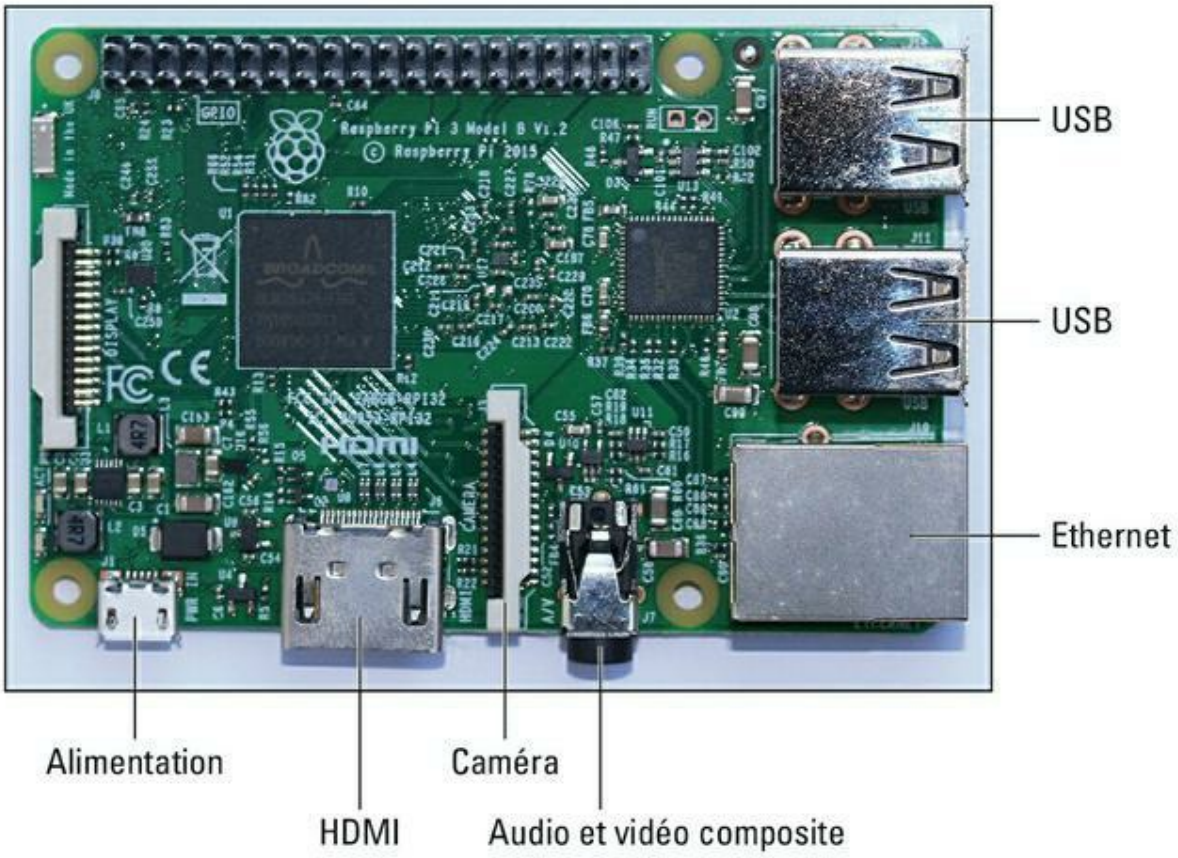


Figure 3.2 : Connexions de la Raspberry Pi 3 et B (valable aussi pour le Pi 2, le modèle B et le modèle A+).

La Figure 3.3 montre les connexions de la Raspberry Pi Zéro W. La Raspberry Pi Zéro ne s'en distingue que par l'absence de connecteur pour la caméra du côté droit.

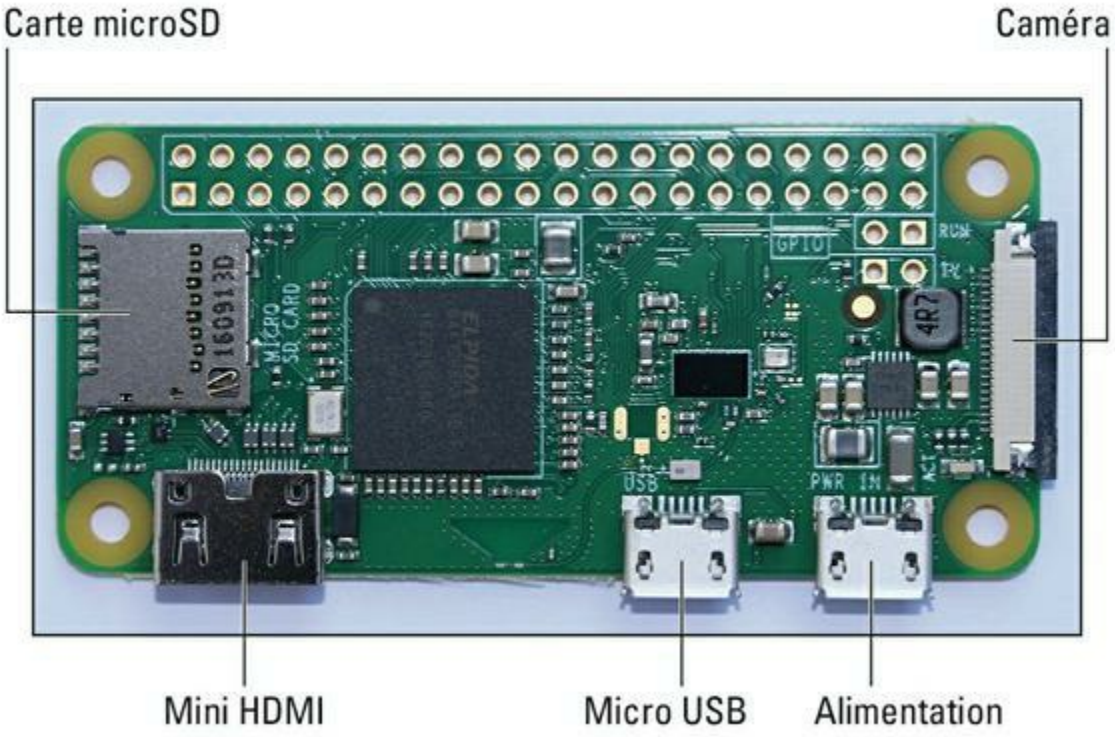


Figure 3.3 : Connexions de la Raspberry Pi Zéro W.

Nous avons indiqué dans le Chapitre 2 les accessoires dont vous aurez besoin pour utiliser votre Raspberry Pi, et notamment les différents câbles.

Insertion de la carte SD

Pour fonctionner, une carte Raspberry Pi a besoin de trouver dans son lecteur de carte mémoire une carte au format SD ou MicroSD selon le modèle, sur laquelle a été installé un système d'exploitation. Si vous n'avez pas encore cette carte, revenez au Chapitre 2 pour apprendre comment télécharger le système et le mettre en place sur la carte mémoire.

Dans le cas d'un modèle 2, 3, A+ ou B+, vous insérez la carte microSD en retournant votre carte Raspberry, comme le montre la Figure 3.4. Vous voyez le lecteur de carte mémoire du côté gauche. Glissez votre carte microSD dans le lecteur de sorte que les contacts viennent toucher les petits ressorts. Avec un modèle Pi 3, la carte va se mettre en place d'elle-même. Dans les autres modèles, un petit clic se fera entendre. La carte doit un peu dépasser, ce qui permettra de l'extraire.

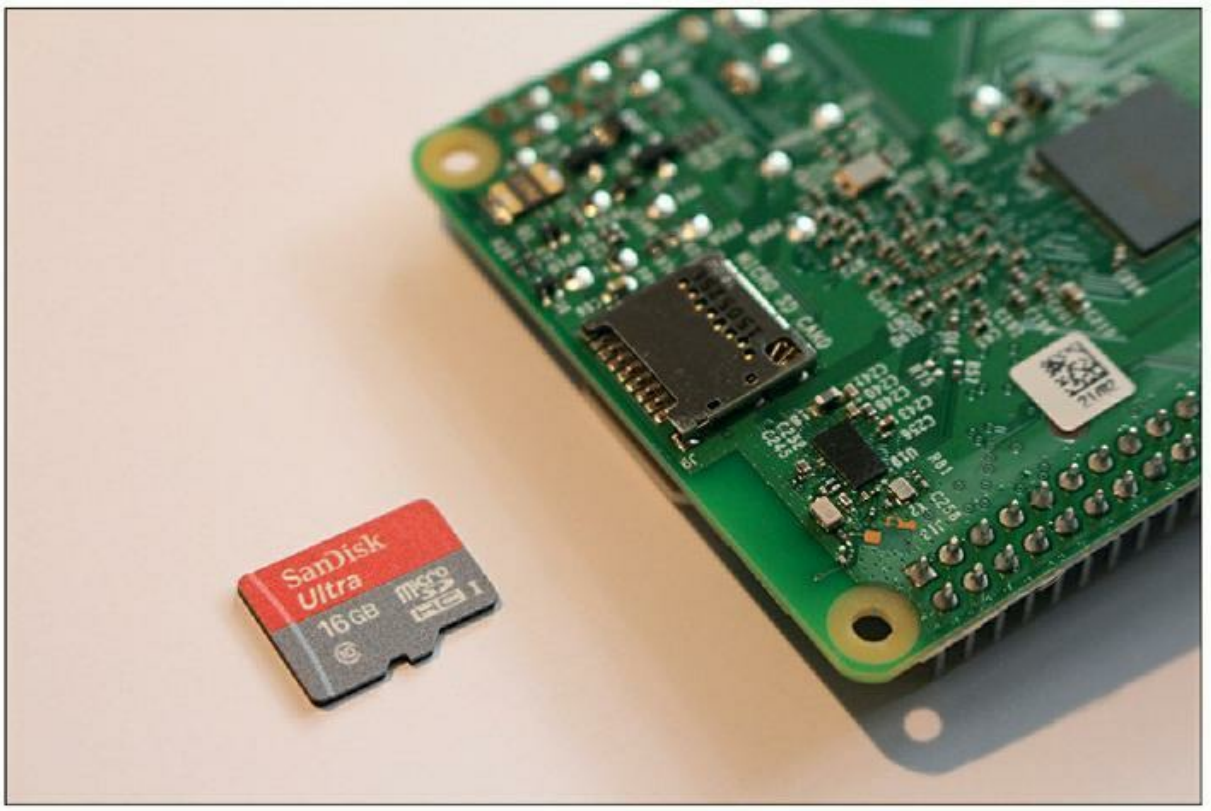


Figure 3.4 : Insertion de la carte microSD dans le modèle Pi 3.

Dans le cas d’un modèle A ou B, pour bien insérer la carte, retournez le circuit (voir Figure 3.5). Sur un des petits côtés, vous repérez un connecteur plat pour carte mémoire. Insérez la carte avec le côté étiquette face à vous (donc à l’envers lorsque vous retournerez le circuit) et enfoncez gentiment la carte à fond. Il est normal que la carte dépasse du circuit et reste visible quand vous aurez retourné celui-ci.

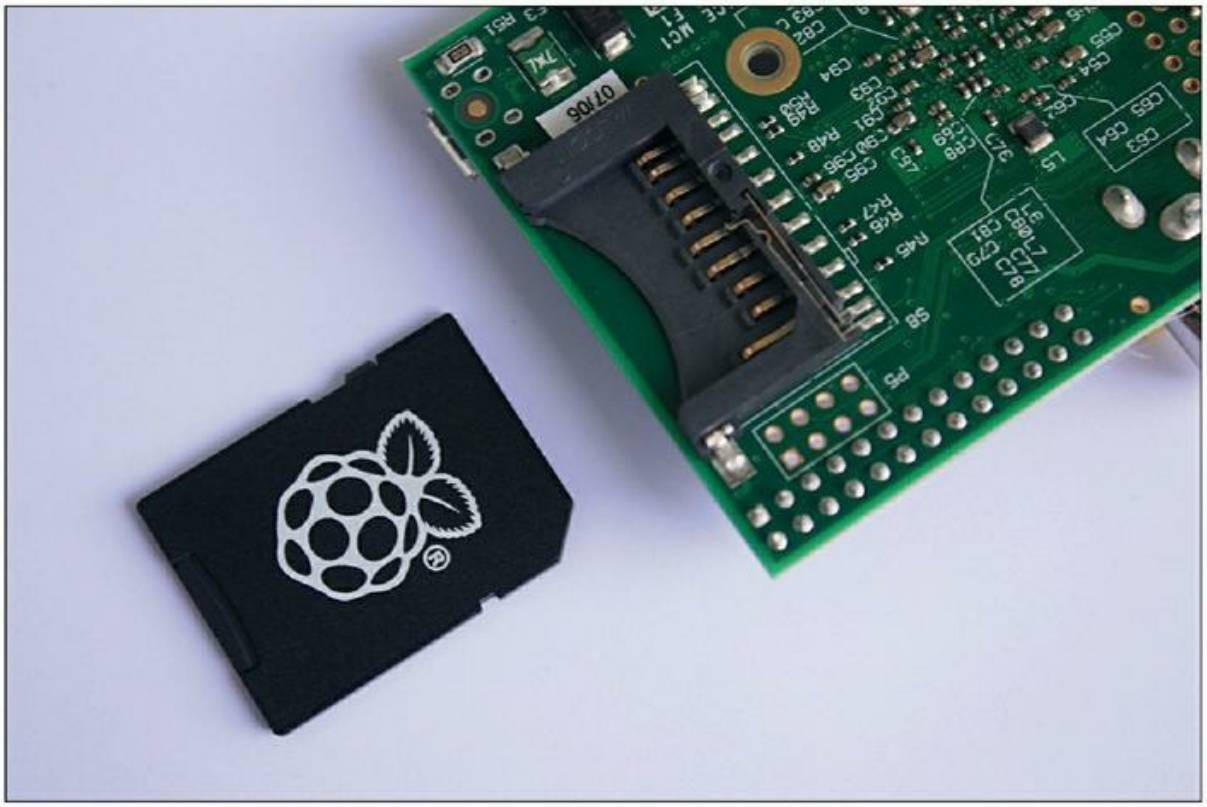


Figure 3.5 : Insertion de la carte au format SD standard dans une carte modèle A.

Pour les deux modèles Pi Zéro, le lecteur de carte microSD est soudé sur le dessus. Insérez votre carte avec l’étiquette face à vous, comme le montre la Figure 3.6.

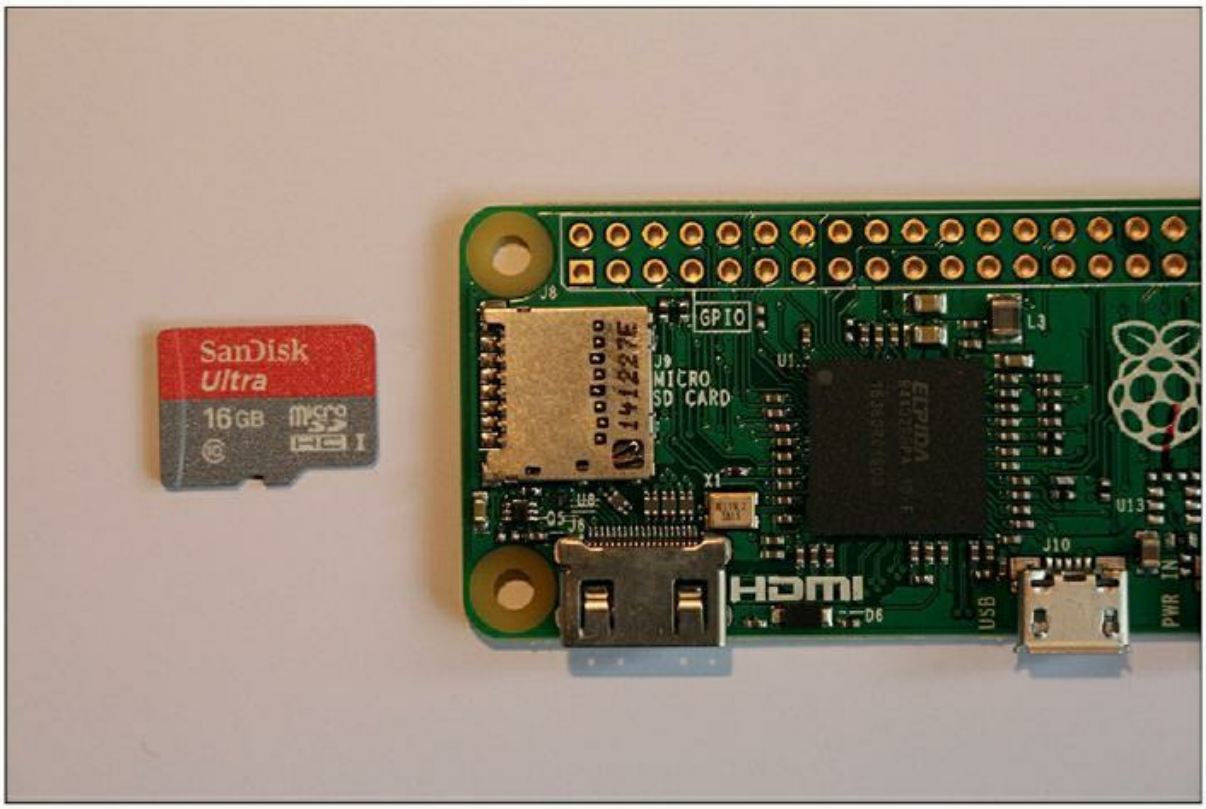


Figure 3. 6 : Insertion de la carte microSD dans un Raspberry Pi Zéro.

Pour retirer la carte, assurez-vous que votre circuit soit hors tension. Pour le modèle Pi 3 ou Pi Zéro, il suffit de tirer doucement. Pour les autres modèles, il faut pousser un peu pour déverrouiller le ressort, en prenant soin de retenir la carte qui pourrait s'éjecter et disparaître sous un meuble.

Nous répétons qu'il ne faut insérer ou enlever une carte mémoire que lorsque l'ensemble du circuit est hors tension sous peine de perdre des données ou d'altérer la carte mémoire.

Branchement du module caméra Raspberry Pi

Il existe un grand nombre d'accessoires et d'extensions pour les cartes Raspberry Pi. Le module Caméra officiel est un des rares accessoires proposés par la fondation. Si vous ne disposez pas de cette caméra, vous pouvez ignorer cette section. Si nous indiquons dès à présent comment la mettre en place, c'est parce que l'opération est beaucoup plus simple lorsqu'elle est réalisée avant d'effectuer les autres branchements. La connexion de la caméra utilise un câble blanc plat, dit *câble en nappe* (Figure 3.7).

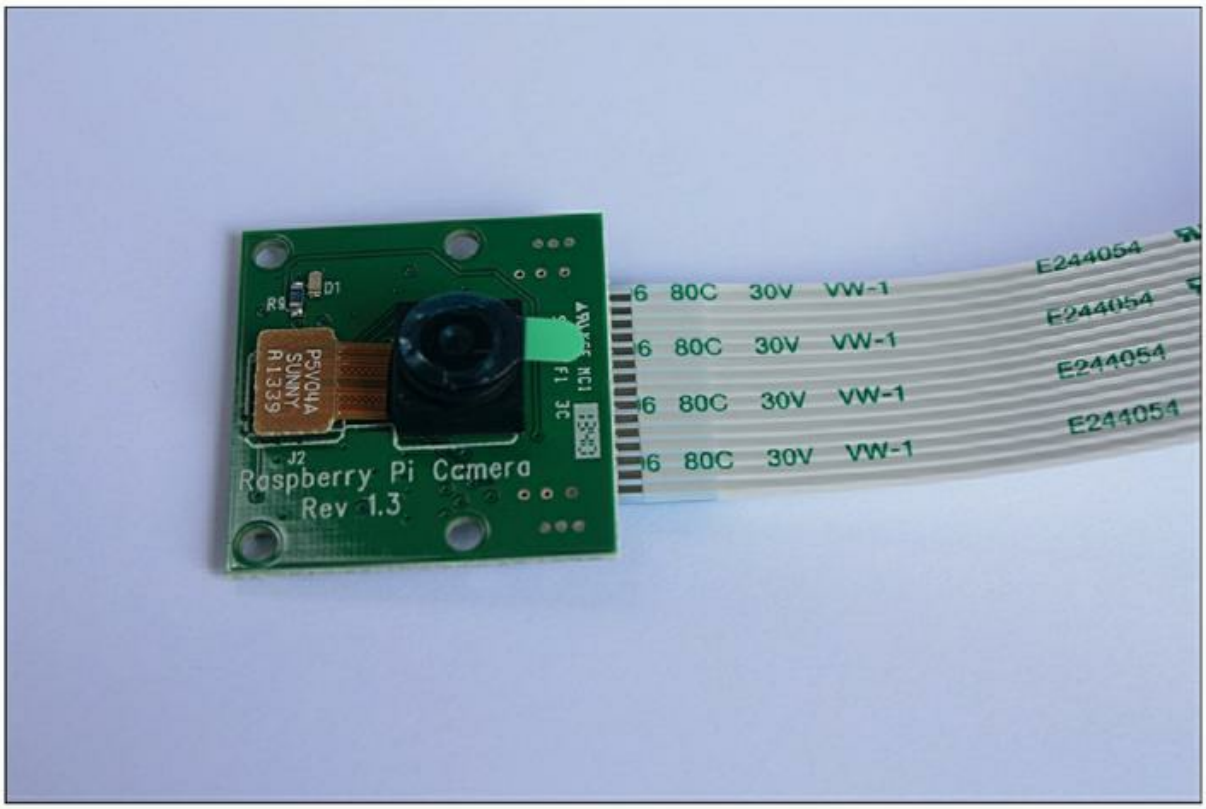


Figure 3.7 : Le module Caméra Raspberry Pi.

Repérez le connecteur de caméra sur votre RasPi. Il est indiqué dans les Figures 3.1, 3.2 et 3.3. Rappelons qu'il n'y a pas de connecteur de caméra sur les Pi Zéro.

Observez bien la lentille de la caméra : elle est livrée avec un film plastique de protection qu'il faut enlever en tirant sur le petit onglet vert.

Une fois la caméra en place, nous devrons effectuer des tests, ce que nous ferons tout à la fin du chapitre.

Branchement de la caméra sur une Pi Zéro W

Le connecteur pour la caméra pour Pi Zéro W utilise un câble qui n'a pas la même largeur que celui des autres modèles de cartes. Vous pouvez acheter ce câble à part ou l'obtenir en même temps que le boîtier officiel pour Pi Zéro.

La carte et la caméra utilisent le même connecteur. Il suffit de déplacer l'étrier du connecteur entre le pouce et l'index puis de tirer légèrement pour l'entrouvrir. Les deux parties ne vont pas se séparer, mais cela va former un espace suffisant pour pouvoir y glisser le câble en nappe. Du côté RasPi, le connecteur se trouve du côté droit (Figure 3.3).

Du côté de la caméra, vous insérez le câble avec les contacts vers l'avant de la caméra puis vous repoussez l'étrier pour reverrouiller le tout. Sur la variante Zéro W, les contacts doivent faire face au dessous de la carte, le côté plat. Quand le câble repose à plat, la caméra doit regarder vers le bas, mais vous pouvez ensuite plier doucement le câble pour que l'objectif regarde vers le haut. Un des boîtiers vendus pour Pi Zéro prévoit un trou pour l'objectif.

Branchement de la caméra sur les autres modèles

Vous devez d'abord ouvrir le verrou du connecteur sur la carte RasPi en tenant les deux extrémités entre le pouce et l'index puis en tirant doucement. Cela permet d'ouvrir l'espace pour y insérer le câble en nappe.

Le câble se termine par des contacts argentés. Le côté des contacts doit regarder vers la gauche, et non du côté des prises USB. Poussez le câble doucement dans le connecteur de façon perpendiculaire, puis repoussez l'étrier pour verrouiller le tout.

Adaptation des Pi Zéro et Pi Zéro W

Les deux cartes Pi Zéro utilisent des connecteurs mini-HDMI et microUSB. Vous devez prévoir des adaptateurs qui permettent de se brancher à une prise HDMI et USB standard. La Figure 3.8 montre un exemple d'adaptateur microUSB vers USB et mini-HDMI vers HDMI.

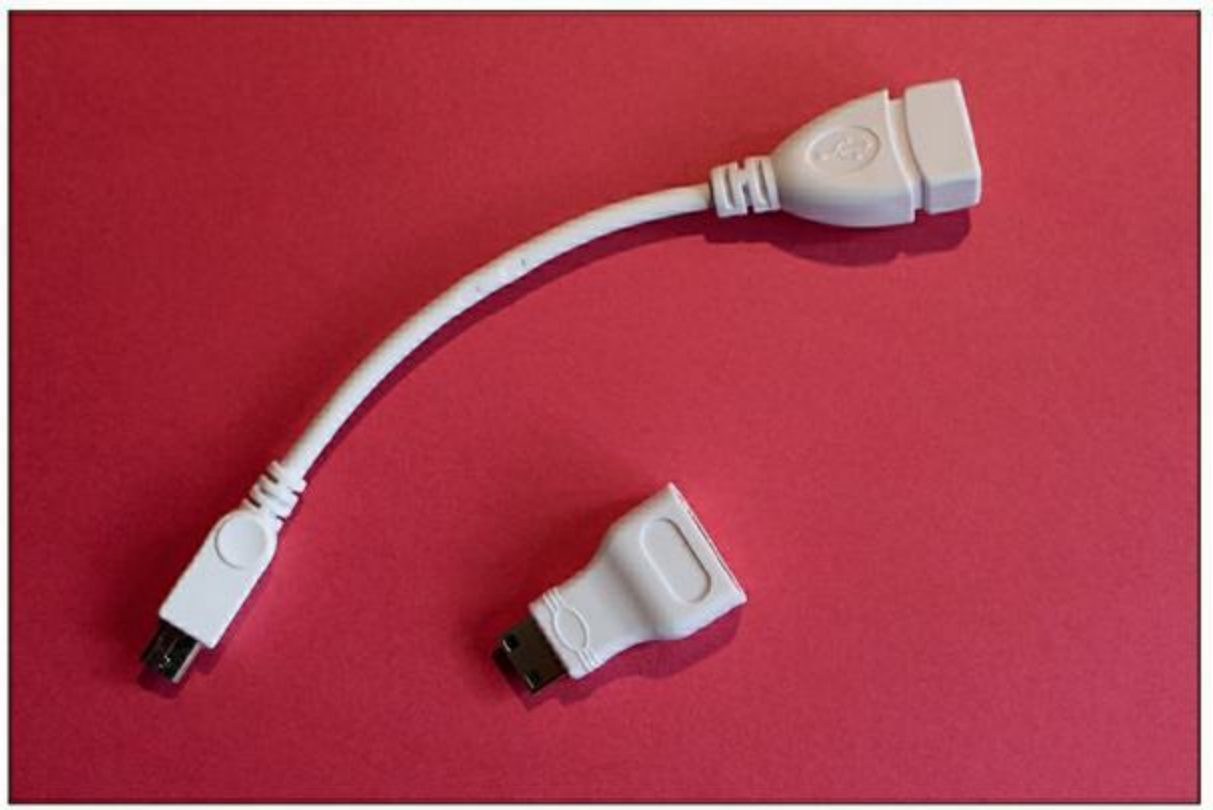


Figure 3.8 : Adaptateur microUSB vers USB et mini-HDMI vers HDMI pour Pi Zéro.

Insérez l'adaptateur HDMI dans la prise mini-HDMI et l'adaptateur USB dans la prise microUSB (indiquée par une mention USB sur la carte). Ne vous trompez pas, car cet adaptateur peut également se brancher dans la prise d'alimentation.

Branchement d'un écran ou d'un téléviseur

Les cartes RasPi proposent deux sorties vidéo. L'une des deux reste donc inutilisée en fonction du type d'écran dont vous disposez.

Nous supposons dans la suite que vous allez utiliser un écran informatique ou un téléviseur. Notez qu'il existe un écran tactile officiel pour Raspberry Pi qui se branche sur la prise Display du côté gauche (non disponible sur les minicartes Pi Zéro).

Branchement d'un écran HDMI ou DVI

Repérez le connecteur HDMI en vous aidant des Figures 3.1 à 3.3. Pour un Raspberry Pi Zéro, vous devez disposer de l'adaptateur mini-HDMI vers HDMI. Insérez le câble HDMI dans le connecteur puis insérez l'autre extrémité dans l'écran.

Si votre écran ne possède qu'une entrée DVI, vous devez ajouter un adaptateur qui est une sorte de prise à brancher au bout du câble HDMI. Pensez à verrouiller l'adaptateur à l'arrière de l'écran au moyen des deux vis moletées. La Figure 3.9 montre l'adaptateur juste avant d'y insérer le connecteur HDMI.

Figure 3.9 : Installation de l'adaptateur DVI au bout d'un câble HDMI.

Branchement d'un téléviseur composite

Si votre écran n'a ni entrée HDMI, ni entrée DVI, vous pourrez peut-être vous en servir dans l'ancien mode vidéo composite. Dans les anciens modèles A et B, le connecteur correspondant est jaune et argent, comme le montre la Figure 3.1.

Sur les modèles récents, Pi 2, Pi 3 et B+, le connecteur est le même que la prise casque. Vous devez vous procurer un câble RCA spécial, car un câble audio ne convient pas.

Branchez votre câble RCA à la prise et l'autre extrémité à l'entrée Video in de votre téléviseur, normalement de couleur jaune.

Vous aurez sans doute besoin de changer la source de signal de votre téléviseur grâce à sa télécommande. D'ailleurs, même en utilisant une entrée HDMI, il faut sans doute allumer l'écran d'abord pour que le Raspberry Pi en détecte les caractéristiques.

Si vous avez choisi le système NOOBS, celui-ci va essayer d'abord d'afficher les images sur un écran HDMI, même si aucun câble n'est connecté. Pour l'obliger à envoyer ses affichages vers le câble RCA, il faut accéder aux options de récupération de NOOBS puis frapper la touche 3 pour le mode d'affichage PAL répandu en Europe. Si vous avez un écran HDMI récent, le simple fait de le brancher le fera reconnaître. Dans le cas contraire, il faut allumer le RasPi, attendre quelques secondes, puis maintenir enfoncée la touche Maj pendant environ 10 secondes lorsque cela vous est demandé. Après cette attente, vous pouvez frapper le chiffre indiqué plus haut pour les options de récupération. Il n'est pas très facile de frapper le chiffre au bon moment, d'autant que rien n'est affiché à l'écran ! L'un des auteurs a essayé plusieurs fois avant de se résoudre à brancher un écran HDMI en même temps que son téléviseur.

Les modèles miniatures Zéro et Zéro W n'ont pas de sortie vidéo composite sur prise dédiée, mais le signal est disponible. Si nécessaire, vous pouvez souder un connecteur sur la carte à l'endroit où vous voyez la légende TV. Dans ce cas, servez-vous des instructions à l'adresse suivante :

`www.raspberrypi.org/magpi/rca-pi-zero`

Branchement d'une multiprise USB (hub)

Les prises USB de votre RasPi sont disposées du côté droit du circuit (Figures 3.1 et 3.2). Comme déjà dit, les cartes miniatures Zéro ont besoin d'un adaptateur microUSB vers USB. Votre multiprise USB, également appelée concentrateur ou hub, n'a besoin que de se brancher *via* un câble USB à l'une des prises disponibles.

Il est essentiel d'utiliser une multiprise USB autoalimentée, donc dotée de son adaptateur secteur.

La Figure 3.10 donne un exemple de multiprise USB fonctionnant avec les cartes RasPi. Le câble USB qui en sort se branche sur une des prises USB de la carte et les autres prises sont disponibles pour vos périphériques. Vous devez voir un petit trou sur la face avant de la

multiprise pour y brancher le jack de l'alimentation. Le modèle présenté offre deux prises de chaque côté, mais il en existe à sept prises !

Figure 3.10 : Multiprise USB compatible Raspberry Pi.

Branchement d'un clavier et d'une souris

Sur les modèles récents comme le modèle B+, les Pi 2 et Pi 3, vous pouvez brancher directement votre clavier et votre souris sur deux des prises USB. Pour les cartes anciennes, nous conseillons fortement de brancher ces périphériques à la multiprise USB autoalimentée pour éviter de tirer trop de courant de la carte RasPi.

Pour les Pi Zéro, le modèle A et le modèle A+, vous êtes forcés d'utiliser une multiprise USB puisqu'il n'y a qu'une seule prise sur la carte. Notez que vous pouvez éviter ces branchements si vous disposez de périphériques Bluetooth. Dans ce cas, voyez la section ultérieure de ce même chapitre qui donne des conseils pour configurer la liaison Bluetooth.

Branchements audio

Si vous avez utilisé le branchement HDMI pour l'affichage, le son est transmis sur ce câble et vous n'avez pas besoin de câble audio séparé.

Dans le cas contraire, vous utiliserez le connecteur audio qui est le petit connecteur bleu ou noir sur le bord supérieur à côté de la sortie vidéo composite sur les modèles A et B (Figure 3.1) et du côté inférieur de la carte pour les modèles B+, Pi 2 et Pi 3 (Figure 3.2). Si vous disposez d'un casque, vous pouvez directement le brancher dessus.

Pour plus de confort, vous brancherez le câble audio vers un téléviseur, une chaîne stéréo ou des haut-parleurs amplifiés. Le câble approprié est présenté dans la Figure 3.5 : d'un côté, il y a deux prises RCA gauche et droite, et de l'autre, un jack stéréo 3,5 mm. Les deux prises RCA correspondent à la plupart des entrées audio des chaînes stéréo. Votre câble sera peut-être différent en fonction du type de connecteur d'entrée de votre équipement audio.

Rappelons que si vous utilisez des haut-parleurs de PC, il faut que ce soit des haut-parleurs amplifiés.

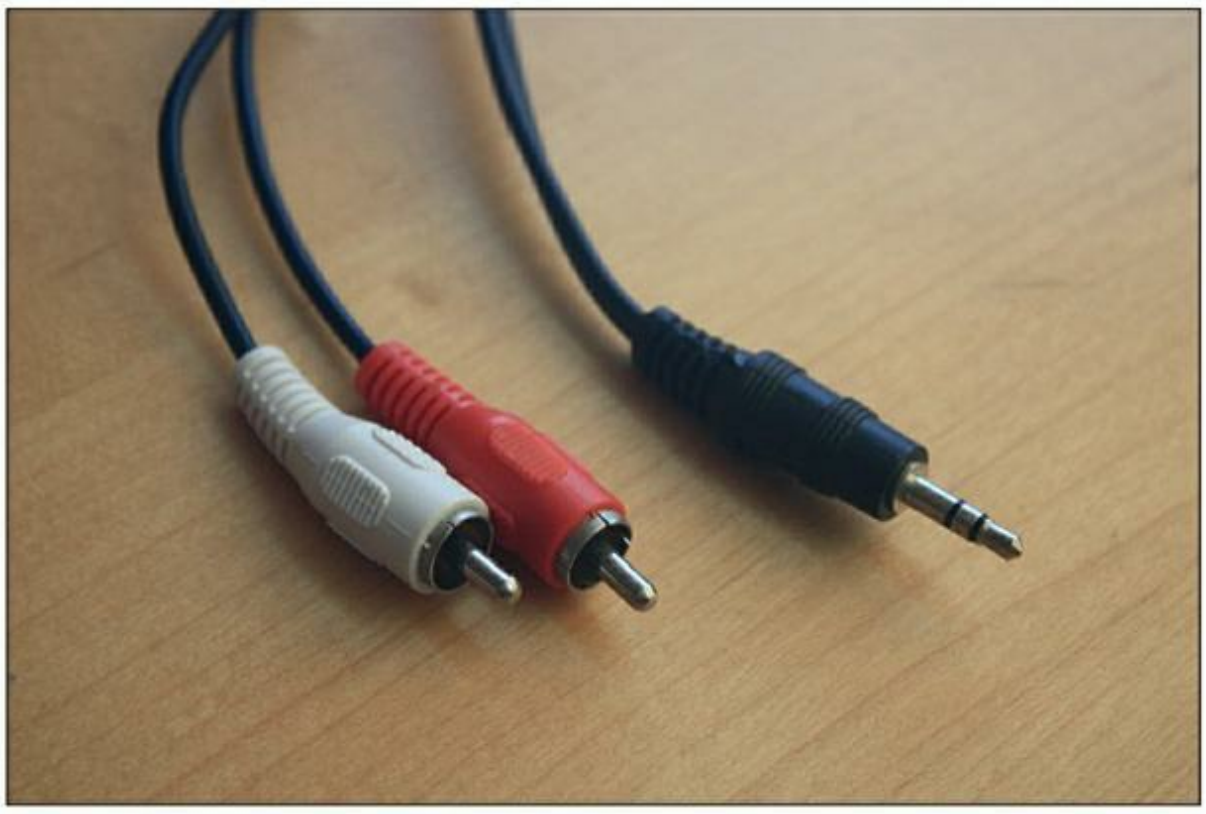


Figure 3.11 : Câble audio de connexion stéréo.

Branchement au routeur

Les cartes modèles A, A+ et Zéro n'ont pas de connexion réseau filaire alors que les autres cartes ont une prise Ethernet du côté droit

(Figure 3.1 et 3.2). Vous pouvez brancher votre carte RasPi à votre routeur Internet avec un câble Ethernet standard.

Si votre routeur utilise le protocole DHCP (*Dynamic Host Configuration Protocol*), qui est activé dans la plupart des cas, le Raspberry Pi saura se connecter automatiquement à Internet. En cas de souci au niveau de la connexion à Internet, vous vous reporterez à l'Annexe A.

Si vous disposez d'un adaptateur Wi-Fi, vous pouvez le brancher dans une prise USB pour être prêt à configurer votre liaison Wi-Fi dès la mise sous tension du RasPi.

Branchement de l'alimentation

Il ne reste plus qu'à brancher l'alimentation secteur. Vous pouvez voir la prise d'alimentation dans les Figures 3.1 à 3.3.

La fondation Raspberry Pi déconseille l'alimentation des cartes par des piles, à moins de bien s'y connaître. Il est en effet assez facile de provoquer des dégâts si vous ne fournissez pas une tension de 5 V bien régulée. Certains chargeurs de téléphone peuvent être utilisés, mais vous devez faire des essais en restant vigilant.

Il n'y a pas d'interrupteur sur le Raspberry Pi ; il démarre dès que vous branchez l'alimentation. Pour l'éteindre, vous devez donc débrancher cette prise. Le fils de l'auteur a pris soin de relier son concentrateur USB et son Raspberry Pi à une multiprise avec interrupteur, ce qui permet d'allumer et d'éteindre le tout d'un seul geste. C'est de loin préférable à l'insertion et à l'extraction répétées de la prise mini-USB qui peut fragiliser les soudures. Cet interrupteur général permet également de se protéger contre les retours de courant (lorsque la multiprise USB renvoie du courant dans la carte RasPi, ce qui peut provoquer des dégâts en cas de surtension).

Si vous avez opté pour la version Lite de NOOBS, il est indispensable d'avoir établi une connexion réseau pour pouvoir télécharger le système que vous allez installer. Si vous avez accès à une liaison filaire Ethernet, branchez le câble réseau. Sinon, vous devez configurer le Wi-Fi, si vous avez une carte qui en est dotée, ou ajouter un adaptateur Wi-Fi dans un port USB. Au démarrage de NOOBS, il vous sera demandé d'effectuer la configuration. Vous pourrez revenir aux paramètres à ce moment en cliquant le bouton Wi-Fi ou en frappant la touche W.

Démarrage

Dès que vous mettez votre RasPi sous tension, vous voyez apparaître un arc-en-ciel puis le menu de sélection du système à installer, en fonction de ce qui a été trouvé sur la carte mémoire (Figure 3.12). Ne négligez pas en bas d'écran les deux listes permettant de choisir la langue et le type de clavier.

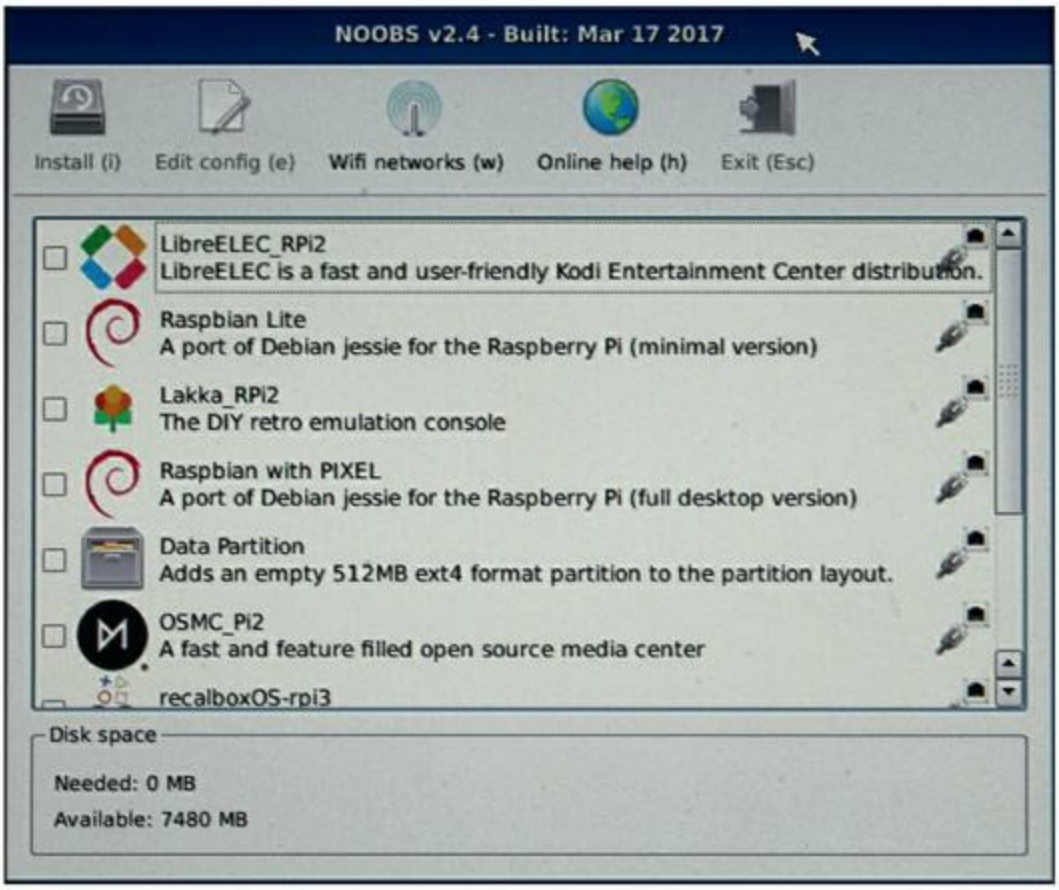

NOOBS v2.4 - Built: Mar 17 2017
Install (i)
Edit config (e)
Wifi networks (w)
Online help (h)
Exit (Esc)
LibreELEC_RPi2
LibreELEC is a fast and user-friendly Kodi Entertainment Center distribution.
Raspbian Lite
A port of Debian jessie for the Raspberry Pi (minimal version)
Lakka_RPi2
The DIY retro emulation console
Raspbian with PIXEL
A port of Debian jessie for the Raspberry Pi (full desktop version)
Data Partition
Adds an empty 512MB ext4 format partition to the partition layout.
OSMC_Pi2
A fast and feature filled open source media center
recalboxOS-rpi3
Disk space
Needed: 0 MB
Available: 7480 MB

Figure 3.12 : Les différents systèmes proposés par NOOBS.

Pour installer un ou plusieurs systèmes, il suffit de cocher la case correspondante. Voici les options proposées dans NOOBS Light :

» **Raspbian**. C'est la distribution Linux préconisée par la fondation Raspberry Pi. Elle est basée sur la distribution Debian, après optimisation pour les circuits Raspberry Pi. Elle est dotée d'une interface graphique, décrite dans le Chapitre 4, d'un navigateur Web et d'un certain nombre d'outils de développement et de programmation. Ce système constitue le choix idéal pour démarrer rapidement dans la découverte de RasPi. Dans toute la suite du livre, sauf dans un chapitre, nous

supposons que vous avez choisi ce système Raspbian. Il en existe deux versions : une dotée de l'interface graphique PIXEL et l'autre beaucoup plus légère, appelée Raspbian Light. Nous vous conseillons d'utiliser la version complète, à moins d'avoir des besoins particuliers.

- **LibreELEC et OSMC**. Ce sont deux variantes du système de centre multimédia Kodi qui permet de regarder des films et d'écouter des musiques. Nous en parlons dans le Chapitre 8.
- **RISC OS**. La grande majorité des propriétaires de cartes RasPi vont l'utiliser sous Linux. Il existe cependant d'autres systèmes, et notamment celui nommé RISC OS, lui aussi doté d'une interface graphique. RISC OS est né en 1987, lorsque la société britannique Acorn Computers l'a adopté pour faire fonctionner son ordinateur familial Archimedes. Le système continue à être maintenu par la société RISC OS Open Limited. À l'heure où nous écrivons ces lignes, RISC OS n'est pas disponible pour le modèle Raspberry Pi 3, mais vous pouvez télécharger l'image et préparez une carte mémoire, comme indiqué à la fin du Chapitre 2.
- **Partition de données data**. Cette option réserve un espace sur la carte mémoire pour y créer une partition de stockage de données qui permet ensuite d'échanger des données entre plusieurs distributions Linux.
- **Lakka**. Il s'agit d'un système d'exploitation pour pouvoir jouer à d'anciens jeux vidéo, c'est-à-dire

une sorte de système de *retrogaming*, en français « nostaljeux ». Il réunit plusieurs simulateurs permettant de redonner vie aux jeux de toute une gamme d'anciens ordinateurs familiaux, et notamment le Commodore 64, l'Amstrad CPC, plusieurs machines Atari, sans oublier le ZX Spectrum. S'y ajoutent des émulateurs pour les consoles de jeux de Nintendo et la Sony PlayStation et l'émulateur MAME (*Multi Arcade Machine Emulator*) qui permet de rejouer à des jeux de salles d'arcade. C'est à vous de vous procurer les fichiers exécutables des jeux que vous avez envie d'essayer, car le système n'est fourni qu'avec les deux jeux *2048* et *Bomberman*. De nombreux anciens jeux ont été rendus disponibles par leurs auteurs gratuitement sur le Web. Pour l'émulateur MAME, voyez à l'adresse suivante :

`http://mamedev.org/roms`

Voyez aussi les jeux pour Amstrad réunis par l'auteur Sean à cette adresse :

`www.sean.co.uk/books/amstrad/index.shtm`

Pour profiter d'un jeu avec Lakka, la solution la plus simple consiste à stocker les fichiers du jeu sur une clé USB puis à insérer cette clé dans votre RasPi. Dans les menus, vous utilisez les flèches du curseur, la touche Entrée pour sélectionner et la touche de retour arrière pour reculer. Dans le système Lakka, le système d'émulation est appelé noyau ou Core, et chaque jeu est appelé contenu. On commence donc par l'option **Load Content**.

Choisissez l'option **Start directory** pour naviguer jusqu'à la clé USB contenant le ou les jeux. Vous devez alors choisir quel noyau utiliser pour émuler le jeu. Si vous utilisez MAME, frappez une des touches 5, 6, 7 ou 8 pour choisir de jouer avec de un à quatre joueurs puis validez par Entrée ou saisissez un autre chiffre entre 1 et 4 pour choisir le nombre de joueurs. Pour jouer, utilisez les flèches du curseur. La touche de tir varie selon le jeu. Essayez l'une des touches Z, X, espace et les touches Ctrl, Alt et Maj pour le joueur un. Vous revenez au menu principal avec la touche Echap. Notez que le système Lakka utilise de nombreuses touches pour contrôler directement les paramètres de fonctionnement, par exemple la vitesse de l'émulateur. Pour reconfigurer les touches, revenez au menu principal, utilisez la flèche droite pour aller aux paramètres, choisissez **Input** puis l'option **Input Hotkey Binds**.

» **Recalbox**. Cet autre système de nostaljeux regroupe des émulateurs pour les consoles SNES, NES, Game Boy Advance, Sega Master System et PC Engine. Il contient également la version en partagiciel du fameux jeu de survie *Doom*. Tous les émulateurs sont accompagnés de versions de démonstration de certains jeux, ce qui suffit à passer une bonne après-midi. Servez-vous des flèches du curseur pour naviguer dans les menus, de la touche S pour confirmer et de la touche A pour revenir en arrière. (*N. d. T. :* parfois, il faut utiliser la touche correspondante sur un clavier QWERTY. Faites des essais.) Les touches à utiliser

sont souvent indiquées en bas d'écran. Si votre clavier possède une touche Windows, servez-vous-en pour accéder aux paramètres. Vous lancez un jeu avec la touche Entrée. En général, vous utilisez les touches du curseur pour vous déplacer dans le jeu et les touches Z et X pour tirer ou sauter. Vous quittez avec Echap. Les deux systèmes Lakka et Recalbox sont compatibles avec la manette de jeu USB créée par la société Pi Hut. Elle ressemble à une manette SNES classique avec les flèches sous le pouce gauche et les quatre boutons ABXY sous le pouce droit.

» **Screenly Open Source Edition (OSE)**. Ce système est dédié à l'affichage sur écran publicitaire, par exemple pour de la publicité dynamique. Il permet de programmer l'affichage d'images, de vidéos et de pages Web sur un écran HD, par exemple dans une galerie commerciale, une vitrine de magasin ou un établissement scolaire.

» **Windows 10 IoT Core**. Ce système ne prétend pas rivaliser avec le système de bureau Windows tel que vous le connaissez. C'est une version de Windows conçue pour gérer des objets interconnectés de l'Internet des objets (*Internet Of Things*). Lorsque nous avons installé ce système, il nous a été demandé de choisir entre la version RTM, prête à être dupliquée, et une préversion. La version RTM étant normalement plus stable, nous l'avons choisie. Il a fallu beaucoup de temps à notre RasPi pour démarrer avec Windows IoT. À certains moments, nous avons même cru que tout

s'était bloqué. Gardez donc patience. Pour créer des programmes, vous devez disposer de l'atelier de développement Microsoft Visual Studio sur une machine sous Windows. Vous transférez ensuite l'exécutable du programme vers le RasPi par une connexion Wi-Fi, Ethernet ou Internet Connection Sharing. Le système est livré avec un tutoriel pour vous aider à faire vos premiers pas. Pour en savoir plus, vous visiterez la page www.windowsondevices.com.

» **TLXOS**. Il s'agit d'une version d'essai du logiciel client léger ThinLinX qui permet d'utiliser un RasPi en tant que bureau virtuel, dialoguant avec un logiciel qui s'exécute sur une autre machine. Le logiciel de gestion permet de gérer de façon centralisée plusieurs machines RasPi. Ce système simplifie donc la gestion de plusieurs machines déployées par exemple pour de l'affichage numérique ou pour du travail en groupe. À la fin de la période d'essai, vous devrez vous acquitter d'une licence pour chaque machine, au prix de 10 €.

Dans le menu de sélection, vous pouvez cocher plusieurs systèmes, la seule condition étant que la carte mémoire offre assez d'espace pour les installer tous en même temps. Vous vérifiez cela dans le bas de la fenêtre. Pensez à garder un peu d'espace pour la partition de données partagées *data* si vous en avez besoin.

Pour basculer l'interface du système en français, ouvrez la liste déroulante tout en bas de l'écran (Figure 3.13) et choisissez **Français**. Normalement, la configuration du clavier de la seconde liste va se mettre à jour automatiquement.

Figure 3.13 : Liste de sélection du français dans NOOBS.

Une fois que vous avez sélectionné le ou les systèmes installés, cliquez le bouton **Install** en haut à gauche.

Dans tous les cas, sélectionnez au minimum le système Raspbian avec PIXEL, car c'est celui que nous utilisons dans toute la suite du livre.

Nous rappelons que le fait d'installer un système efface tout le contenu antérieur de la carte mémoire.

Si vous avez choisi de travailler avec la version allégée NOOBS Light, les systèmes vont alors être téléchargés puis implantés sur la carte. Si vous utilisez la version complète, autonome, de NOOBS, Raspbian et les outils système sont installés directement depuis le contenu actuel de la carte.

Une fois l'installation terminée (soyez patient), un message vous le confirmera. Si vous n'avez installé que Raspbian, la carte va redémarrer avec ce système.

Si vous avez installé plusieurs systèmes, il vous sera demandé au démarrage de choisir avec lequel vous voulez démarrer. Double-cliquez dans votre choix. Si vous ne choisissez rien, le système redémarre avec le dernier système utilisé.

Si vous avez envie de réinstaller un ou plusieurs systèmes à partir de NOOBS, il suffit de redémarrer en maintenant enfoncée la touche Maj. Rappelons que cette opération efface tout le contenu antérieur de la carte.

Ouverture de session (login)

Lorsque vous mettez votre Raspberry Pi sous tension, vous devez fournir un nom d'utilisateur (un identifiant) et un mot de passe. Les données à fournir varient en fonction de la version de Linux que vous avez choisi d'utiliser. Si vous avez opté pour la distribution standard Raspbian Wheezy, le nom du premier utilisateur est toujours et le mot de passe est .

Notez bien que si vous n'avez pas choisi un clavier français dans la configuration, vous devrez taper la touche **Q** pour avoir le **A** dans le mot de passe **raspberry**.

Aussi bien l'identifiant que le mot de passe distinguent les majuscules des minuscules. Vous ne pouvez donc pas saisir **PI** en capitales. Notez que rien n'est affiché pendant la saisie du mot de passe, pas même le fait que vous ayez saisi un certain nombre de lettres, ce qui peut décontenancer au départ. Saisissez avec soin et vous devriez réussir à vous connecter.

Vous confirmez que vous avez réussi à vous connecter au système en voyant apparaître l'invite de la ligne de commande suivie d'un caractère de soulignement clignotant :

```
pi@raspberrypi ~ $
```

Cela signifie que votre Raspberry Pi est prêt à recevoir vos ordres. Vous pouvez saisir des commandes Linux pour gérer vos fichiers et lancer vos programmes. (Le Chapitre 5 vous fait entrer dans les arcanes du système Linux en vous montrant comment utiliser les principales commandes pour piloter le Raspberry Pi et sa montagne de fichiers.)

Configuration du système Raspbian

Dans toute la suite du livre, sauf dans le Chapitre 8, nous utilisons le système Raspbian avec son interface graphique PIXEL. C'est le système le plus facile à utiliser, et donc le plus apte à vous aider à faire vos premiers pas avec Raspberry Pi.

Vous devez voir apparaître l'interface graphique dès que la machine a terminé de démarrer. Nous découvrirons cette interface dans le prochain chapitre. Pour l'instant, voyons comment terminer la configuration initiale.

Cliquez l'icône du menu général dans le coin supérieur gauche de l'écran. Accédez à la catégorie **Préférences** puis choisissez **Configuration Raspberry Pi** pour obtenir la boîte de dialogue de la Figure 3.14.

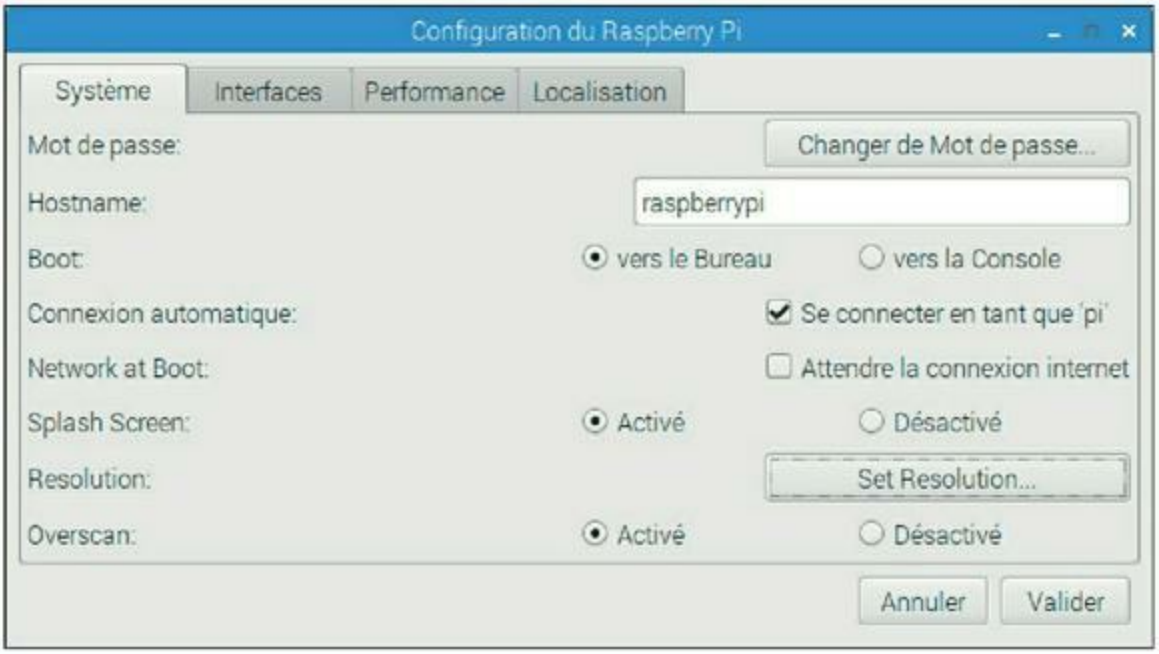


Figure 3.14 : Boîte de configuration de Raspbian avec PIXEL.

Au départ, vous voyez apparaître la première page, **Système**. Voici les options disponibles :

- **Changement du mot de passe**. Le mot de passe initial pour l'utilisateur **pi** est **raspberry**. Notez que si le mot de passe ne semble pas être reconnu, c'est que vous avez peut-être laissé le clavier en anglais, auquel cas il faut taper la touche Q au lieu de la touche A.
- **Nom de la machine hôte**. C'est le nom sous lequel cette carte Raspberry Pi est connue dans le réseau local.
- **Choix du démarrage** avec l'interface graphique, c'est-à-dire le bureau, ou en mode texte, avec une console de type terminal. Nous en parlons dans le Chapitre 5.
- **Connexion automatique** de la session avec l'utilisateur **pi**.

> Une option permet d'attendre que le réseau soit accessible pour démarrer.
>
> L'option **Splash screen** fait afficher un écran d'accueil au démarrage.
>
> Le bouton de résolution permet d'optimiser la résolution d'affichage de l'écran.
>
> Enfin, l'option **Overscan** permet de retoucher avec précision l'alignement de l'image avec l'écran. Désactivez cette option si vous voyez un cadre noir et activez-la si le bureau ne semble pas occuper tout l'espace d'affichage disponible.

Les options de la deuxième page, **Interfaces**, servent à activer certaines options de communication de votre RasPi. C'est ici que vous devez activer l'utilisation de la caméra. D'autres options sont destinées à la connexion à distance sécurisée par SSH, qui permet de prendre le contrôle de la machine à distance. L'option RealVNC permet même de piloter le RasPi à distance avec une interface graphique. Les autres interfaces disponibles gèrent d'autres périphériques *via* le connecteur GPIO selon les protocoles SPI, I2C, Serial, 1-One et Remote GPIO. Vous n'activerez chacune de ces options que si le projet sur lequel vous travaillez a besoin d'utiliser le protocole de communication correspondant.

La page **Performances** permet d'augmenter la fréquence d'horloge et de modifier la proportion de mémoire dédiée au coprocesseur graphique GPU.

L'augmentation de fréquence d'horloge (*overclocking*) consiste à faire fonctionner un processeur plus vite que ce que préconise son fabricant. Si des choix vous sont proposés ici, c'est que la fondation Raspberry Pi a réalisé des tests prouvant que les choix offerts ne risquent pas d'écourter la durée de vie de votre carte. La vitesse du processeur se mesure en millions de hertz (MHz). La fréquence la plus élevée qu'il est possible d'atteindre est de 1 000 MHz. Les choix réellement disponibles dépendent du modèle de carte et de la qualité de l'alimentation. Notez que la carte Raspberry Pi 3 ne permet actuellement pas d'augmenter la fréquence.

La mémoire vive (RAM) de votre Raspberry Pi est utilisée en mode partagé par le processeur central (CPU) et le coprocesseur graphique (GPU). Les deux processeurs sont nécessaires pour exécuter les

programmes, mais certains programmes ont plus besoin de calculs par le CPU alors que d'autres ont plus besoin de traitements d'images, qui sont faits par le GPU. Si vous comptez vous servir de la machine surtout pour lire des vidéos et exécuter des jeux en 3D, vous augmenterez les performances en attribuant un plus grand espace relatif au GPU. Inversement, vous augmenterez les performances des traitements généraux en diminuant l'espace réservé au GPU pour la restituer au CPU. La distribution Raspbian prévoit au départ 64 Mo pour le GPU, tout le reste allant au CPU. Cette configuration convient dans la plupart des cas, mais si vous avez des problèmes de vitesse d'affichage, vous pouvez intervenir. L'option vous demande la quantité de mémoire que vous désirez attribuer au coprocesseur GPU et vous propose une valeur dans la zone de saisie. Tout le reste de la mémoire sera attribué au processeur central. Vous pouvez faire des essais et voir quel est le meilleur équilibre en fonction du type d'application dont vous avez surtout besoin.

La dernière page de la boîte de configuration, **Localisation**, est extrêmement importante pour les francophones. C'est ici que vous choisissez la langue dans laquelle sont présentés les menus et la configuration du clavier (si vous ne l'avez pas fait lors de l'installation depuis NOOBS). Ouvrez la première liste déroulante et choisissez **France** (Figure 3.15).

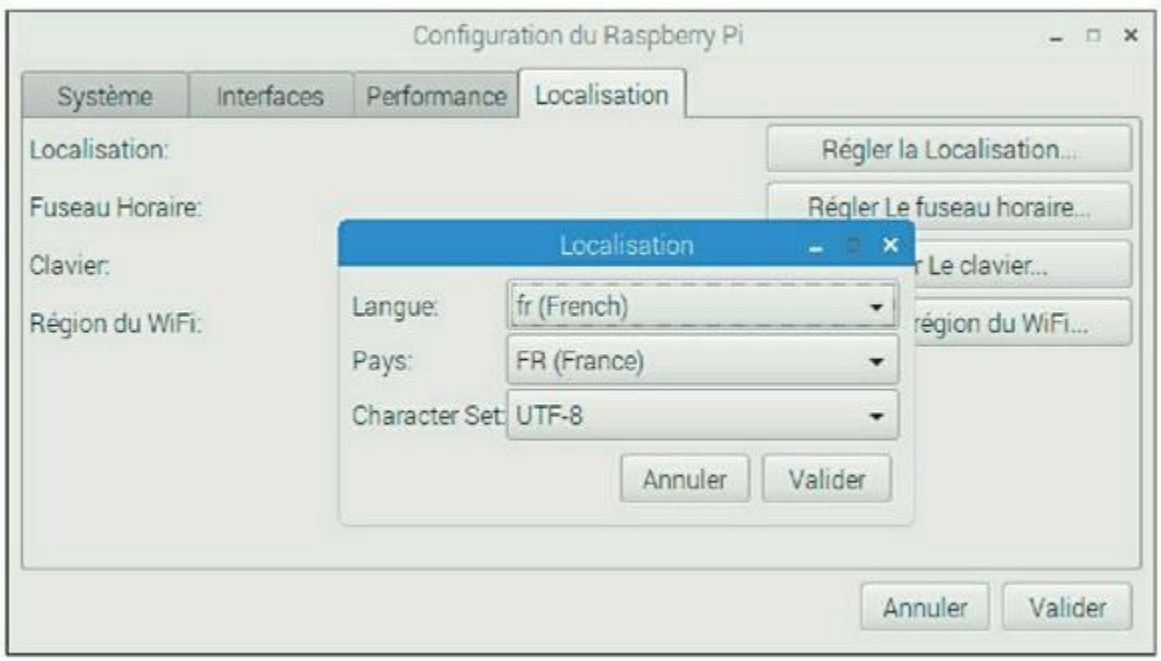


Figure 3.15 : Choix des paramètres de localisation.

Ouvrez ensuite les autres listes pour vérifier et modifier éventuellement le réglage afin de disposer du fuseau horaire Europe et de la ville Paris (ou un autre choix selon votre localisation).

Vous pouvez régler la réactivité de la souris du clavier au moyen d'un module de la catégorie Préférences du menu général.

Si vous avez choisi d'utiliser Raspbian sans interface graphique, vous accédez aux mêmes options de configuration en tapant la commande **sudo raspi – config**. Bien sûr, vous ne pouvez pas utiliser la souris, mais vous disposez des flèches du curseur pour choisir les options et pour sélectionner celle-ci. La touche Tab permet aussi de sélectionner une des actions OK, Cancel, Select et Finish. Vous confirmez par la touche Entrée.

Configuration Wi-Fi

Si votre carte possède un module Wi-Fi, vous le configurez en cliquant l'icône située juste à droite de celle du Bluetooth, tout en haut à droite de l'écran.

Un menu apparaît avec les différents réseaux accessibles, et une option permet de désactiver le Wi-Fi.

Figure 3.16 : Les boutons Bluetooth et Wi-Fi (en haut à droite).

Dans la figure précédente, vous pouvez distinguer tout à fait à droite un triangle souligné. Ce bouton est extrêmement important : lorsque vous avez inséré une clé USB ou un disque externe, c'est ici que vous devez cliquer pour déconnecter logiquement le périphérique, ce qu'il faut absolument faire avant d'extraire mécaniquement la clé ou le disque.

Les codes d'accès aux réseaux Wi-Fi sont de nos jours suffisamment complexes pour qu'il soit intéressant de s'épargner leur saisie. Une solution simple consiste à préparer le code d'accès à votre réseau Wi-Fi sur un PC ou un Mac en le stockant dans un petit fichier texte. Vous placez ensuite ce fichier texte sur une clé USB (dans la racine) et vous insérez la clé USB dans votre RasPi. Lorsque le système vous demande si vous voulez ouvrir le contenu, vous répondez favorablement puis vous double-cliquez le nom du fichier texte pour pouvoir copier le code.

Lorsque vous ouvrez la liste des réseaux accessibles, sélectionnez celui que vous désirez et collez le code dans la boîte qui apparaît. Lorsque la connexion Wi-Fi est impossible, le symbole de Wi-Fi est remplacé par deux traits verticaux barrés par deux croix rouges.

Pour tester la connexion, il suffit d'ouvrir le navigateur Web (voir le Chapitre 4).

Le système mémorise la configuration de la connexion Wi-Fi. Il est inutile de la reconfigurer au prochain démarrage, ou après avoir désactivé et réactivé le Wi-Fi. Cela reste valable même si vous avez configuré la connexion en mode ligne de commande.

En revanche, si vous changez de réseau, le RasPi oublie la configuration précédente.

Lorsque vous avez terminé d'utiliser la clé USB pour récupérer le code d'accès Wi-Fi, pensez à la déconnecter logiquement (la démonter) en utilisant l'icône de triangle dans l'angle supérieur droit du bureau.

Configuration Bluetooth

La carte Raspberry Pi 3 et la Pi Zéro W disposent d'un module Bluetooth, ce qui permet de travailler avec un clavier et une souris sans fil. Cela dit, il vous faut au départ une souris et un clavier USB pour installer le système. Une fois que vous êtes arrivé au niveau du Bureau graphique, vous pouvez configurer vos périphériques Bluetooth.

Ce n'est pas parce qu'un clavier ou une souris est « sans fil » qu'il est Bluetooth. Les périphériques qui sont livrés avec leurs propres adaptateurs USB utilisent en général une autre norme de connexion sans fil.

Pour que deux appareils Bluetooth entrent en communication, il faut les apparier. Voyez les instructions fournies avec votre matériel pour que celui-ci puisse être détecté par la carte RasPi. Ce n'est pas toujours simple. La souris d'un des deux auteurs l'a obligé à enfoncer et maintenir un bouton puis à enfoncer les deux boutons de la souris en même temps jusqu'à ce que la souris commence à clignoter.

Une fois que votre périphérique est détectable, cliquez l'icône Bluetooth en haut à droite (Figure 3.16) et choisissez d'ajouter un périphérique, Add device. Dès que la carte a trouvé l'objet, cliquez puis utilisez le bouton Pair. Lorsque nous avons configuré un clavier, il nous a fallu saisir un code affiché à l'écran. Pour une de nos souris, le système nous a même

demandé de saisir le code affiché sur celle-ci, alors que la souris n'a pas d'écran. Nous avons simplement confirmé et tout s'est bien passé.

Test du module Caméra

Cette section ne vous concerne que si vous avez ajouté le module caméra Raspberry Pi. Nous allons dans ce cas voir si elle fonctionne.

Pour que la caméra puisse être testée, il faut que le canal de communication correspondant soit activé. Revenez dans l'outil de configuration Raspberry Pi décrit quelques pages plus haut puis accédez à la page Interface et choisissez **Activer** dans la ligne de la caméra.

Nous allons effectuer les tests sur la ligne de commande en mode texte (mode décrit en détail dans le Chapitre 5). Ouvrez une fenêtre de terminal au moyen de l'icône de terminal en haut d'écran (un rectangle noir contenant les deux symboles >_). Voici comment essayer de prendre une photo :

```
raspistill -o testshot.jpg
```

Vous devriez voir à l'écran ce que voit la caméra juste avant que la photo soit prise. Pour vérifier que le fichier image a été généré, affichez la liste du contenu du répertoire courant :

```
ls
```

De nombreuses options sont disponibles pour prendre des photos. Voici par exemple comment ajouter un filtre de couleur pastel tout en retournant horizontalement l'image puis verticalement :

```
raspistill -ifx pastel -hf -vf -o testshot2.jpg
```

Tous ces tirets et toutes ces lettres isolées peuvent sembler étranges, mais les choses deviendront bien plus claires lorsque vous aurez terminé la lecture du Chapitre 5. Vous pouvez même afficher la documentation en ligne de cette commande :

```
raspistill 2>&1 ¦ less
```

Vous vous déplacez dans le texte au moyen de la flèche Bas du curseur et vous sortez de l'affichage avec la touche Q.

A priori, vos photos sont stockées dans votre répertoire de travail *pi*. Le Chapitre 4 indique comment utiliser le Gestionnaire de fichiers pour trouver vos fichiers et l'outil de visionnage pour les visualiser.

Voyons maintenant comment capturer une vidéo. La commande suivante permet de capturer un film de cinq secondes :

```
raspivid -o testvideo.h264 -t 5000
```

Pour visualiser la vidéo, vous utilisez l'outil suivant :

```
omxplayer testvideo.h264
```

Le fichier est généré au format brut H264. Pour une meilleure compatibilité avec les lecteurs multimédias, nous conseillons d'effectuer une conversion vers le format MP4. Il faut d'abord installer l'outil MP4Box :

```
sudo apt-get install gpac
```

Vous pouvez ensuite lancer la conversion d'un fichier vidéo. Dans notre exemple, le nom est *testvideo.h264*. Le résultat est un fichier appelé *testvideo.mp4* :

```
MP4Box -add testvideo.h264 testvideo.mp4
```

Pour de l'aide au sujet du module vidéo, saisissez cette commande :

```
raspivid 2>&1 ¦ less
```

Vous disposez en outre d'une librairie de fonctions Python qui permet donc d'exploiter la caméra depuis un programme. Pour tout détail au sujet du module caméra officiel, visitez la page suivante :

www.raspberrypi.org/documentation/usage/camera

Configurer la partition data

Lors de l'installation du ou des systèmes au niveau de NOOBS, il vous a été proposé de mettre en place une partition de données partagées, ce qui permet d'échanger des données entre deux systèmes.

Cette partition porte obligatoirement le nom *data*. Si vous l'avez fait installer, il ne vous reste plus qu'à créer une liaison logique entre cette

partition de la carte et un sous-répertoire vide de votre système. Pour ce faire, il est préférable d'avoir déjà un peu d'expérience avec le système Linux. Si ce n'est pas le cas, nous vous conseillons de faire vos gammes en expérimentant la totalité du Chapitre 5 avant de revenir ici procéder à cette connexion logique.

1. **Démarrez votre carte RasPi avec le système Raspbian puis ouvrez une fenêtre de terminal pour accéder à la ligne de commande**.

2. **Depuis votre répertoire principal, créez un sous-répertoire avec la commande suivante :**

```
mkdir monrep
```

3. **Saisissez la commande suivante pour effectuer la connexion logique entre la partition et ce répertoire, c'est-à-dire le montage :**

```
sudo mount -L data monrep
```

4. **Saisissez la commande suivante pour vous donner suffisamment de droits d'accès afin de pouvoir écrire dans ce dossier :**

```
sudo chown $USER: monrep
```

5. **Vous pouvez maintenant entrer dans ce dossier avec la commande suivante :**

```
cd monrep
```

Tous les fichiers entreposés dans ce dossier seront accessibles aux autres systèmes que vous pouvez démarrer depuis la même carte mémoire.

Prêt pour l'aventure ?

Votre carte Raspberry Pi est maintenant configurée et prête à fonctionner. Par où poursuivre ?

Le Chapitre 4 présente l'environnement graphique du bureau avec ses fenêtres et ses icônes, son navigateur Web, son Gestionnaire de fichiers, sa visionneuse d'images, son éditeur de texte, *etc*.

Quant au Chapitre 5, il vous emmène dans les coulisses du système Linux en vous montrant comment maîtriser l'utilisation du système en mode texte sur la ligne de commande.

PARTIE 2
Découvrir Linux

DANS CETTE PARTIE

- Utiliser l'environnement graphique PIXEL pour gérer les fichiers et démarrer les programmes de votre Raspberry Pi.
- Naviguer sur le Web et créer des favoris.
- Visionner des photos.
- Sauvegarder sur une carte SD.
- Rechercher et installer des applications.
- Explorer le système Linux et maîtriser la structure arborescente des fichiers.
- Utiliser l'interpréteur shell de Linux pour copier, supprimer, déplacer les fichiers sur une carte mémoire SD, gérer les comptes utilisateurs et installer des logiciels.

Chapitre 4

L'environnement graphique

DANS CE CHAPITRE :

- **Naviguer dans l'environnement graphique**
- **Exploiter les périphériques de stockage externes depuis le Bureau**
- **Utiliser le Gestionnaire de fichiers et le visionneur d'images**
- **Naviguer sur le Web**
- **Personnaliser le bureau**

La façon la plus intuitive de partir à la découverte de votre Raspberry Pi consiste à utiliser l'interface graphique qui s'appelle ici PIXEL (*Pi Improved XWindow Environment Lightweight*). PIXEL appartient à la distribution Raspbian préconisée pour les cartes Raspberry Pi, comme déjà indiqué dans le Chapitre 3. C'est cette interface qui apparaît lorsque vous démarrez votre machine. Elle est basée sur une interface graphique plus simple, nommée LXDE (*Lightweight X11 Desktop Environment*), à laquelle la fondation Raspberry Pi a apporté quelques améliorations.

Cet environnement graphique s'utilise d'une façon très proche de celle de Windows ou de macOS. Vous vous servez d'une souris pour désigner des icônes, gérer les fichiers et naviguer dans le système. Une telle interface permet d'unifier les modalités de navigation en les rendant intuitives, ce qui vous encourage à essayer tous les logiciels livrés avec votre distribution Linux, et ceux que vous pouvez installer en complément.

Nous allons voir dans ce chapitre comment trouver vos repères dans l'environnement graphique, et nous vous présenterons une petite sélection des programmes livrés.

Naviguer sur le bureau

La Figure 4.1 montre l'aspect général de l'environnement graphique PIXEL. La photographie qui sert de fond d'écran est peut-être différente sur votre machine.

Tout en haut de l'écran, vous trouvez la barre des tâches, qui reste normalement visible quelle que soit l'application que vous utilisez.

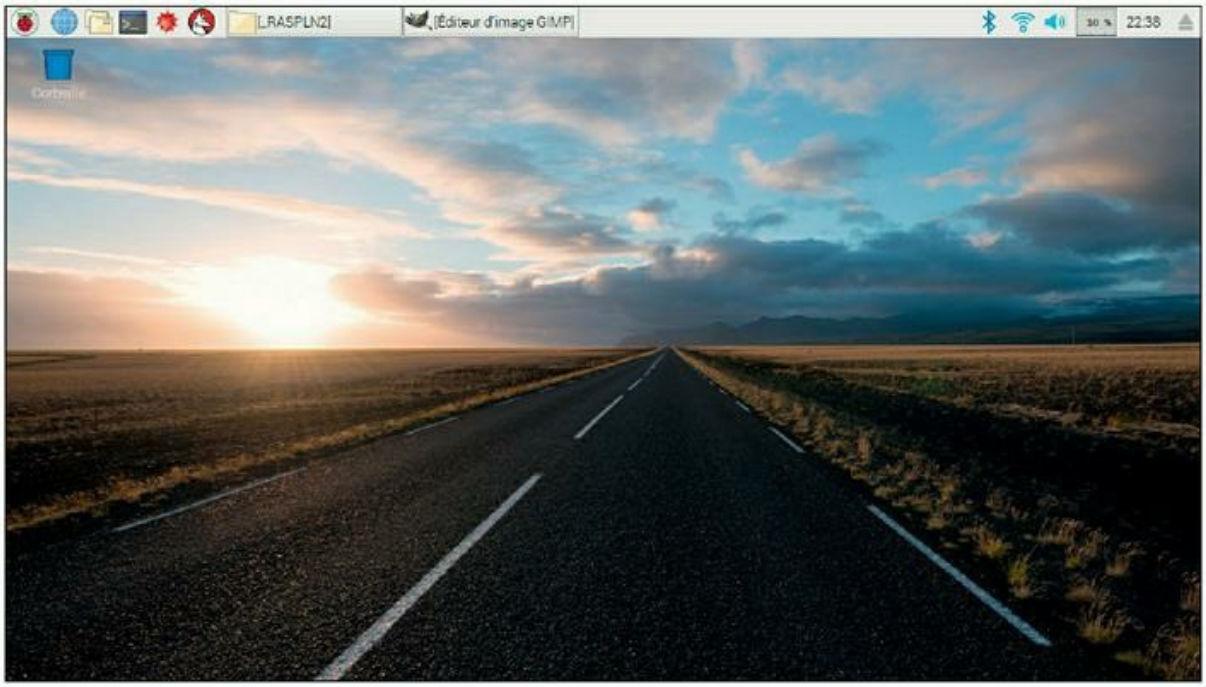


Figure 4.1 : L'environnement graphique PIXEL.

Découverte du menu général

Le menu général est situé dans l'angle supérieur gauche de l'écran ; il représente une framboise. C'est à partir de lui que vous démarrerez la plupart de vos applications. Cliquer cette icône pour voir apparaître les catégories d'application (Figure 4.2).

En amenant le pointeur sur le nom d'une catégorie, le sous-menu correspondant s'ouvre. Il suffit de cliquer pour démarrer l'application désirée.

En cliquant droit dans le nom d'une application, vous pouvez demander d'ajouter une icône d'accès direct sur le bureau.

Figure 4.2 : Sous-menu des accessoires du menu général.

Les différents sous-menus du menu général donnent accès à de nombreux programmes, dont voici une sélection :

» **Claws Mail.** Il s'agit d'un programme de messagerie électronique pour le Raspberry Pi. Nous en parlons un peu plus loin dans ce chapitre. L'application fait partie de la catégorie Internet dans les menus.

» **Guide de référence Debian.** Le système d'exploitation *Raspbian* est une version de Linux Debian spécifique au Raspberry Pi. C'est cette icône qui donne accès à la documentation utilisateur correspondante. Elle est stockée dans des fichiers sur la carte mémoire SD, mais affichée dans le navigateur, comme s'il s'agissait de pages Web. Pour l'instant, la version française n'est pas encore disponible. Vous commencez en double-cliquant dans cette icône puis en choisissant en haut d'écran le lien HTML (*Multi-Files*). Vous n'aurez sans doute pas souvent besoin d'accéder

à ces informations, mais il est bon de savoir qu'elles sont disponibles lorsque vous êtes bloqué.

» **LibreOffice.** Cette suite bureautique complète réunit un traitement de texte (Writer), un tableau (Calc), un présentateur (Impress) et un logiciel de dessin (Draw). Elle est décrite dans le Chapitre 6. Vous y accédez par le menu Bureautique.

» **LXTerminal :** Cette icône provoque l'ouverture d'une fenêtre en mode texte qui permet d'émettre des commandes en direction de l'interpréteur, comme dans le mode texte (voir Chapitre 5), mais sans quitter l'environnement graphique. C'est une technique très pratique lorsque vous voulez lancer une commande pour une opération particulière, tout en restant dans l'environnement PIXEL.

» **Mathematica.** Logiciel de calcul scientifique basé sur le langage Wolfram.

» **Minecraft Pi.** Version spéciale du jeu de construction *Minecraft* pour les cartes Raspberry Pi. Vous pouvez prendre le contrôle du jeu avec le langage Python comme nous le montrons dans le Chapitre 13. L'accès se fait par la catégorie Jeux du menu général.

» **Python 2 et Python 3.** Ce langage est présenté dans les Chapitres 11 et 12. Les conseils qui s'y trouvent permettent de commencer à programmer en Python. Vous trouverez également un nouvel éditeur nommé Thonny IDE qui permet de rédiger et de tester ses

programmes Python. Tous ces outils se trouvent dans la catégorie Programmation.

» **Python Games.** Il s'agit d'une collection de jeux de démonstration créée par Al Sweigart. Vous y trouverez notamment un *Reversi*, un *Puissance 4*, un *Taquin* et un jeu du serpent. L'interface propose un menu dans lequel vous pouvez choisir parmi les treize jeux disponibles.

» **Scratch.** Il s'agit d'un langage visuel d'initiation à la programmation, accessible à tous les âges. Il vous permet de créer des jeux et des animations, entre autres. Nous présentons ce langage dans les Chapitres 10 et 11, en vous montrant comment créer votre propre jeu vidéo.

» **Sense HAT emulator.** Extension qui permet de créer des montages d'expérimentation grâce à une série de capteurs installés sur la carte. Nous en reparlerons dans le Chapitre 19. L'émulateur fait partie de la section Programmation du menu général.

» **Shutdown.** Le nom de cette commande n'est pas traduit en français, mais elle permet d'arrêter votre système. Utilisez toujours cette commande avant de couper l'alimentation. Elle donne accès à un menu dans lequel vous pouvez également choisir de vous déconnecter de votre compte ou de redémarrer la machine. Cette option est au premier niveau dans le menu général.

» **Sonic Pi.** Il s'agit d'un langage de programmation musicale que nous décrivons dans le Chapitre 14. Il fait partie de la catégorie Programmation.

» **Wolfram.** Ce langage de programmation scientifique est destiné aux programmeurs curieux. Vous pouvez en apprendre plus à son sujet à l'adresse www.wolfram.com. L'accès se fait naturellement par la catégorie Programmation.

L'angle supérieur gauche de l'écran (Figure 4.2) comporte quelques autres boutons pour accéder directement à certaines applications : le navigateur Web, le Gestionnaire de fichiers, le Terminal, Mathematica et Wolfram.

Lancer un programme absent des menus

Un certain nombre de programmes n'apparaîtront pas dans le menu des applications, même une fois installés. Pour les démarrer, il suffit d'utiliser l'option d'exécution, comme ceci :

1. **Ouvrez le menu général en cliquant dans la framboise en haut à gauche**.
2. **Choisissez l'option Run ou Exécuter**.
3. **Dans la boîte qui apparaît, saisissez le nom du programme et validez par Entrée**.

Profitez de l'option d'aide à la saisie pour éviter de saisir le nom complet de votre programme. Si vous saisissez par exemple **pengu**, le système propose d'office de finir la saisie à votre place en écrivant **penguinspuzzle**.

Retailler et fermer une fenêtre

Vous aurez certainement envie et besoin de démarrer plus d'un programme à la fois dans votre session PIXEL. Il faut donc savoir comment quitter un programme et gérer les fenêtres pour bien les redisposer dans la surface de l'écran.

Les fenêtres dans PIXEL possèdent des éléments de contrôle proches de ceux de Microsoft Windows pour les retailler et les fermer. En guise

d’exemple, la Figure 4.3 montre l’outil Gestionnaire des tâches qui propose les éléments de contrôle suivants :

- Dans l’angle supérieur droit de la fenêtre de l’application, nous trouvons le classique bouton X pour quitter l’application.
- Juste à côté, nous trouvons le bouton *Maximiser* qui permet de déplier la fenêtre pour qu’elle occupe tout l’écran. Vous pouvez utiliser le même bouton pour restaurer la fenêtre dans sa taille intermédiaire, exactement comme sous Windows.
- À sa gauche, le bouton *Icônifier* (Réduire) permet de replier la fenêtre sans quitter le programme. Vous restaurez l’affichage de la fenêtre en cliquant dans son nom dans la barre des tâches en bas d’écran (elle est aussi appelée *Tableau de bord*).

Vous disposez d’un bouton dans la barre des tâches pour replier d’un geste toutes les fenêtres afin d’avoir un écran vide (revoyez la Figure 4.2).

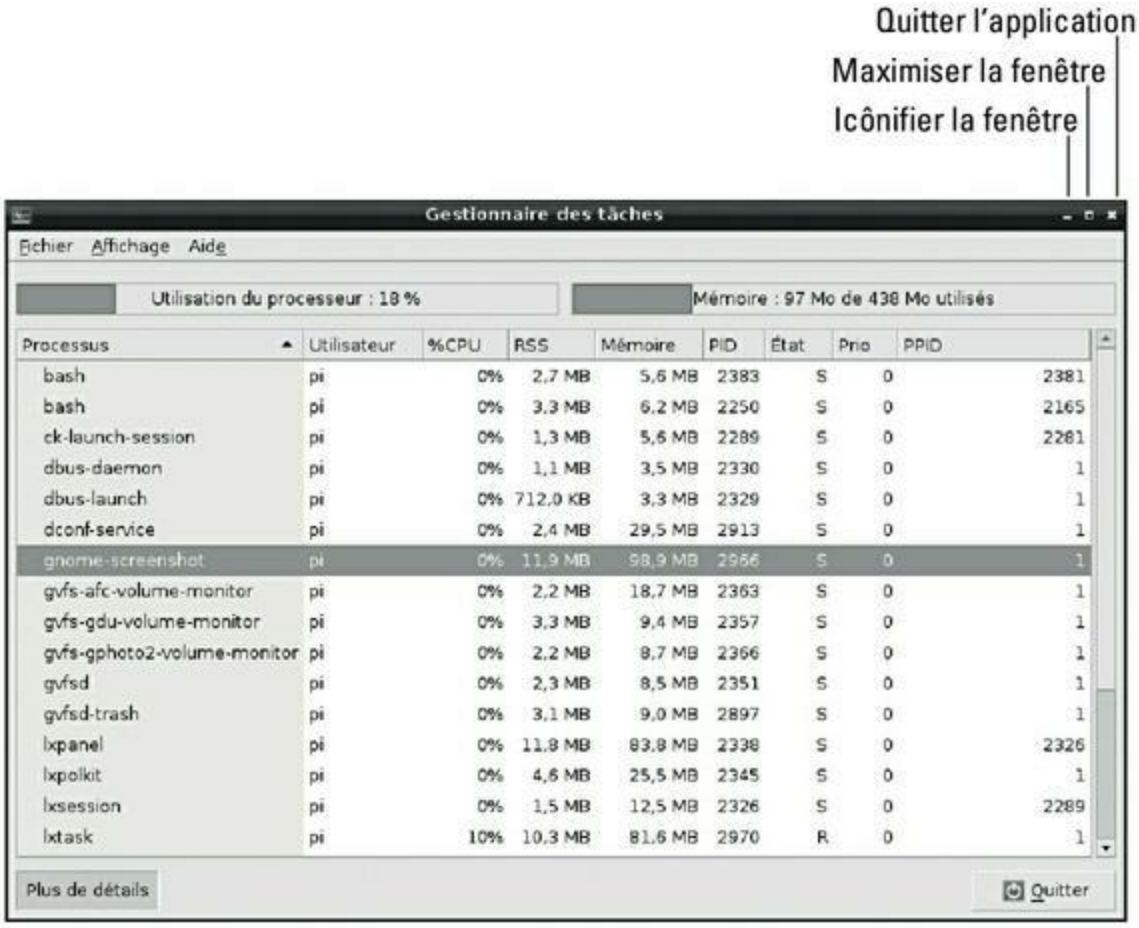

Quitter l'application
Maximiser la fenêtre
Icônifier la fenêtre
Gestionnaire des tâches
Fichier Affichage Aide
Utilisation du processeur : 18 %
Mémoire : 97 Mo de 438 Mo utilisés
Processus Utilisateur %CPU RSS Mémoire PID État Prio PPID
bash pi 0% 2,7 MB 5,6 MB 2383 S 0 2381
bash pi 0% 3,3 MB 6,2 MB 2250 S 0 2165
ck-launch-session pi 0% 1,3 MB 5,6 MB 2289 S 0 2281
dbus-daemon pi 0% 1,1 MB 3,5 MB 2330 S 0 1
dbus-launch pi 0% 712,0 KB 3,3 MB 2329 S 0 1
dconf-service pi 0% 2,4 MB 29,5 MB 2913 S 0 1
gnome-screenshot pi 0% 11,9 MB 98,9 MB 2966 S 0 1
gvfs-afc-volume-monitor pi 0% 2,2 MB 18,7 MB 2363 S 0 1
gvfs-gdu-volume-monitor pi 0% 3,3 MB 9,4 MB 2357 S 0 1
gvfs-gphoto2-volume-monitor pi 0% 2,2 MB 8,7 MB 2366 S 0 1
gvfsd pi 0% 2,3 MB 8,5 MB 2351 S 0 1
gvfsd-trash pi 0% 3,1 MB 9,0 MB 2897 S 0 1
lxpanel pi 0% 11,8 MB 83,8 MB 2338 S 0 2326
lxpolkit pi 0% 4,6 MB 25,5 MB 2345 S 0 1
lxsession pi 0% 1,5 MB 12,5 MB 2326 S 0 2289
lxtask pi 10% 10,3 MB 81,6 MB 2970 R 0 1
Plus de détails
Quitter

Figure 4.3 : Exemple de fenêtres dans PIXEL.

En modifiant la taille des fenêtres, vous optimisez l'occupation de votre écran. Amenez votre pointeur sur un bord horizontal ou vertical jusqu'à ce qu'il prenne un autre aspect. Vous pouvez alors cliquer et déplacer vers l'intérieur ou vers l'extérieur. Si vous faites ce geste dans un angle, vous pouvez modifier les deux dimensions à la fois. Pour déplacer une fenêtre, il suffit de l'agripper en cliquant dans sa barre de titre, en dehors des boutons actifs.

Le Gestionnaire de tâches

Si votre Raspberry Pi semble ne plus réagir, c'est peut-être tout simplement parce qu'il est trop chargé. Vous disposez dans le coin droit de la barre des tâches (tableau de bord) d'un petit outil dont la légende se lit *Moniteur d'utilisation du processeur*. Le graphique donne un aperçu de la charge courante du processeur. Le défilement se fait de droite à gauche. Si une barre verte remplit toute la hauteur, c'est que le Raspberry Pi est franchement très occupé. Prenez l'habitude de patienter un peu, notamment lorsque vous démarrez de grosses applications. De mon expérience, le Raspberry Pi ne se bloque que rarement, mais il peut

facilement être surchargé au point que l'on puisse croire qu'il est bloqué. La patience est la mère des vertus.

Pour savoir quels programmes sont en cours d'exécution, utilisez l'outil **Gestionnaire des tâches** (voir Figure 4.3). Pour le démarrer, accédez au menu général et au sous-menu **Accessoires**. Vous pouvez aussi frapper directement la combinaison de trois touches Ctrl + Alt + Suppr.

Lorsqu'un programme ne répond vraiment plus, vous pouvez forcer sa fermeture grâce à cet outil. Cliquez droit dans sa ligne et choisissez la commande **Terminer**. Cette action envoie une demande de fin au programme, ce qui lui permet de sauvegarder ses données. Si cela ne suffit pas, vous pouvez choisir une mesure plus drastique avec la commande **Tuer (Kill)** qui provoque la fin brutale du programme, avec le risque de perdre les données non enregistrées.

N'utilisez le Gestionnaire des tâches qu'en dernier ressort pour quitter un programme. La plupart des tâches présentées dans la liste de l'outil sont des tâches système qui doivent continuer à s'exécuter pour que PIXEL fonctionne correctement. Ne provoquez pas l'arrêt d'un programme que vous ne reconnaissez pas, car vous risquez de bloquer PIXEL et de perdre les données de toutes les applications en cours.

Le Gestionnaire de fichiers

Il est toujours possible de gérer vos fichiers en émettant des commandes dans l'interpréteur (nous le verrons dans le Chapitre 5), mais il est beaucoup plus confortable de le faire dans l'environnement PIXEL. L'outil nommé Gestionnaire de fichiers (Figure 4.4) permet de tout faire : parcourir les dossiers, copier, supprimer, renommer et gérer les fichiers de la carte SD de votre Raspberry Pi ou des périphériques amovibles que vous aurez insérés.

Le Gestionnaire de fichiers étant un outil indispensable, vous pouvez le démarrer en profitant du bouton de lancement direct de la barre des tâches (revoyez la Figure 4.2). Vous pouvez également plus classiquement ouvrir le menu des programmes et chercher dans le sous-menu **Accessoires**.

Dans le monde Linux, on utilise en général le terme *répertoire* pour le classeur dans lequel sont stockés les fichiers. PIXEL utilise le terme *dossier*, beaucoup plus répandu dans les autres environnements graphiques. Un dossier ou un répertoire permet de regrouper des fichiers de données ou de programmes en attribuant un nom au regroupement. Vous pouvez bien sûr créer des sous-dossiers dans des dossiers.

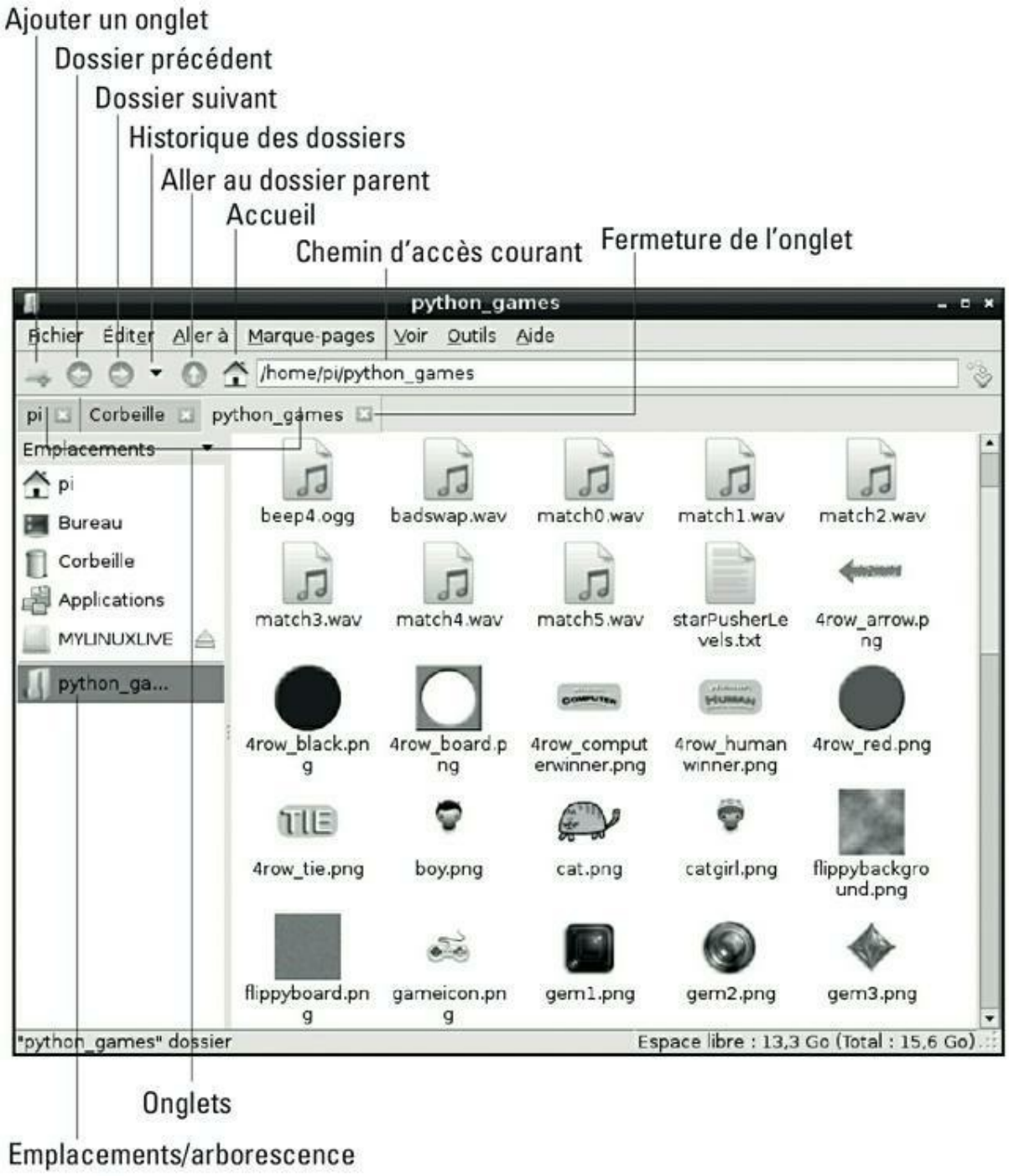


Figure 4.4 : Vue générale du Gestionnaire de fichiers PIXEL sur le Raspberry Pi.

Navigation avec le Gestionnaire de fichiers

La surface principale du Gestionnaire de fichiers (nous abrégerons par Gestionnaire de fichiers) est un panneau servant à afficher le contenu du dossier actuellement sélectionné dans le panneau de gauche. À chaque fichier correspond une icône qui permet d'en deviner le type, sauf pour les fichiers d'images qui bénéficient de l'affichage d'une miniature du

contenu. La Figure 4.4 permet de voir plusieurs types de fichiers concernant les jeux Python livrés en standard avec le Raspberry Pi. Vous pouvez deviner les vignettes des fichiers d'images et les fichiers des effets sonores (une icône contenant deux notes de musique).

Pour accéder à un sous-dossier, il suffit de double-cliquer sur son nom. Pour ouvrir un fichier avec le programme associé par défaut au type correspondant, vous double-cliquez dans son nom aussi. Au départ, les fichiers d'images sont ouverts avec l'outil Visionneur d'images ; un fichier Scratch est ouvert dans l'atelier Scratch. Pour changer le programme associé à un type de fichier, cliquez droit dans l'icône d'un fichier et choisissez l'option **Ouvrir avec...**. Vous voyez apparaître une fenêtre dans laquelle sont présentés tous les programmes installés sur le Raspberry Pi. Il ne vous reste plus qu'à choisir (à bon escient).

Le volet gauche correspond à l'arborescence qui permet d'aller dans n'importe quel sous-dossier. Vous cliquez dans un dossier ici pour faire apparaître son contenu à droite. L'icône représentant un signe moins permet de replier le dossier et donc de cacher ses sous-dossiers. L'icône représentant un signe plus permet de déplier le dossier.

Le dossier nommé **pi** est votre dossier personnel principal, dans lequel vous allez stocker la plupart de vos fichiers et créer vos sous-dossiers (vos documents et vos photos). C'est le seul emplacement dans le système pour lequel vous avez le droit de lire, d'écrire et de modifier les fichiers en gardant le statut d'utilisateur normal. Nous verrons dans le prochain chapitre comment est structuré le système de fichiers Linux. Pour l'instant, retenez qu'il ne faut essayer de stocker des fichiers et créer des sous-dossiers que dans votre dossier personnel *pi* (ou dans un sous-dossier de *pi*).

Le sous-dossier de *pi* nommé *Bureau* à gauche (mais *Desktop* à droite) montre tous les fichiers qui se trouvent actuellement présents sur le bureau sous forme d'icônes. Si vous devez fréquemment revenir à un même document, il suffit de le déposer sur le bureau en le déplaçant dans ce dossier.

Dans l'environnement graphique, lorsque vous insérez un disque dur ou une clé USB, le système va automatiquement le détecter. Il va afficher une boîte de dialogue comme celle de la Figure 4.5 pour vous demander ce que vous voulez faire. En général, vous accepterez de l'afficher dans le Gestionnaire de fichiers. Pour trouver ce périphérique dans l'arborescence de gauche, dépliez les détails du répertoire racine symbolisé par la barre oblique / au moyen du signe plus, puis affichez le contenu du répertoire *media* puis du sous-répertoire *pi*. Avant de débrancher un disque ou une clé USB, vous devez toujours effectuer la déconnexion logique en utilisant le bouton de l'angle supérieur droit (qui représente un triangle).

Cette opération correspond à l'éjection ou démontage. Vous êtes ainsi certain que les données qui n'étaient pas encore écrites sur le support le seront.

Périphérique amovible inseré
Périphérique amovible inseré
Type de périphérique : logiciel Windows
Veuillez sélectionner l'action que vous voulez effectuer :
Ouvrir dans le gestionnaire de fichiers
Annuler
Valider

Figure 4.5 : Fenêtre montrant un périphérique USB externe détecté automatiquement par le Raspberry Pi.

La barre de menus principale du Gestionnaire de fichiers propose les menus **Fichier**, **Éditer**, **Aller à**, **Marque-page**, **Voir**, **Outils** et **Aide**. La plupart des commandes et options qui vous y attendent sont également disponibles par un autre moyen, comme nous allons vous le montrer. Si vous êtes perdu, le menu général est un bon recours pour retrouver ses repères facilement.

Vous pouvez définir un marque-page ou un favori pour un dossier si vous l'utilisez souvent. L'idée s'inspire de ce que permettent les navigateurs Web. Un marque-page permet de retrouver aisément un emplacement déjà visité. Pour créer un marque-page pour le dossier actuellement affiché, ouvrez le menu **Marque-pages** dans la barre de menus du Gestionnaire de fichiers et choisissez **Marquer cette page**. Tous vos marque-pages sont listés dans la partie inférieure du panneau gauche sous le trait séparateur. Il suffit de cliquer dans un nom pour afficher le contenu du dossier correspondant.

Sous la barre de menus, nous trouvons une barre d'icônes qui propose un certain nombre de raccourcis pour les commandes les plus utilisées. Ils sont indiqués dans la Figure 4.5 :

» **Ajouter un onglet :** Les onglets multiples sont très pratiques dès que vous devez travailler sur plusieurs dossiers à la fois. Vous pouvez ainsi rapidement passer entre un dossier source et un dossier destination pour copier des fichiers. Le concept d'onglet est issu des classeurs à onglets qui permettent de passer d'une section à une autre d'un simple geste du pouce. Ils sont devenus la règle dans les fenêtres des navigateurs Web, ce qui permet de passer facilement d'une page Web ouverte à une autre. Le Gestionnaire de fichiers reprend ce principe. Il suffit de cliquer dans le nom d'un onglet en haut de panneau pour en afficher le contenu. La page de chaque onglet est totalement autonome, dans le sens où vous pouvez utiliser toutes les commandes disponibles dans l'outil. Dans la Figure 4.5, plusieurs onglets ont été ouverts : il y en a un pour le dossier personnel **pi**, un pour la Corbeille et un pour le sous-dossier **python_games**. Vous fermez un onglet en utilisant l'icône X à droite du nom d'onglet (voir Figure 4.5).

» **Dossier précédent :** Le Gestionnaire de fichiers mémorise tous les noms des dossiers dans lesquels vous êtes passé, et ce bouton permet de les revisiter en marche arrière, comme dans un navigateur Web. Vous pouvez ainsi revenir jusqu'au premier dossier affiché au démarrage de votre session.

» **Dossier suivant :** Ce bouton devient utilisable à partir du moment où vous avez au moins effectué un déplacement d'un dossier vers un autre. Il

permet de repartir en marche avant dans la liste des dossiers visités.

- **Historique des dossiers :** Ce bouton-menu permet d'accéder à la liste des dossiers visités. Vous pouvez ainsi directement revenir à des dossiers déjà vus, ce qui évite de devoir utiliser à répétition les deux boutons précédents.
- **Aller au dossier parent :** Lorsqu'un dossier est créé dans un autre, il s'agit d'un sous-dossier, et l'autre dossier est son parent. Votre dossier du bureau **Desktop** se trouve dans votre dossier principal **pi** ; ce dossier **pi** est donc le parent du bureau. Ce bouton permet de remonter d'un niveau dans l'arborescence des dossiers. Vous obtenez le même effet au moyen de la touche Ret Arr (celle qui sert normalement à supprimer le caractère à gauche du curseur dans les traitements de texte).
- **Dossier personnel** (*Home*) : Ce bouton vous ramène directement dans votre dossier personnel d'utilisateur, qui est le dossier **pi** pour le compte utilisateur initial.
- **Chemin d'accès** (*path*) : Un chemin d'accès est une description sous forme de texte de l'emplacement d'un sous-dossier avec la liste des dossiers parents qui mènent à lui. Nous présentons en détail le concept de chemin d'accès dans le Chapitre 5. Si vous connaissez un chemin, vous pouvez le saisir dans cette zone puis valider par la touche Entrée pour accéder directement au dossier correspondant.

Copier et déplacer des fichiers et dossiers

Le Gestionnaire de fichiers permet facilement de copier et de déplacer des fichiers et des dossiers, sans avoir besoin de saisir une commande en mode texte.

Lorsque vous cliquez droit dans un nom de fichier ou de dossier dans le Gestionnaire de fichiers, vous voyez apparaître un menu local qui permet de renommer, supprimer, couper ou copier le fichier ou le dossier.

Lorsque vous coupez un élément, il est prêt à être *déplacé* vers l'endroit où vous allez ensuite le coller. Lorsque vous *copiez* un élément, c'est une copie qui sera collée à l'endroit choisi. Pour *coller* un élément, vous vous rendez dans le dossier désiré puis vous cliquez droit à un endroit libre dans la fenêtre pour choisir **Coller** dans le menu local. (Si vous oubliez de coller après avoir coupé un fichier, le fichier de départ est inchangé.)

Vous pouvez utiliser les opérations glisser-déposer pour déplacer des fichiers dans un dossier. Il suffit de relâcher les éléments sur l'icône du dossier.

Sélection multiple

Plusieurs techniques permettent de sélectionner plus d'un fichier à la fois, afin de pouvoir appliquer une opération à un lot d'éléments (suppression, copie ou déplacement) :

- Vous pouvez tout d'abord maintenir enfoncée la touche Ctrl puis cliquer successivement dans chacun des noms de fichiers pour les sélectionner.
- Pour sélectionner une série d'icônes ou de noms consécutifs (de gauche à droite puis de haut en bas), vous cliquez normalement dans la première icône, puis vous maintenez enfoncée la touche Maj tout en cliquant dans la dernière icône de la série.

» Vous pouvez également cliquer dans l'arrière-plan du Gestionnaire de fichiers et maintenir le bouton de souris enfoncé pour capturer par zonage les fichiers à sélectionner.

Une fois que vous avez sélectionné un lot, vous déplacez ce lot dans un autre dossier en cliquant sur un des fichiers de la sélection puis en déposant le tout dans le dossier destinataire. Vous pouvez cliquer droit dans un des noms du lot pour choisir la commande Copier ou Couper (voir Figure 4.6). De même, si vous supprimez un des fichiers sélectionnés, tout le lot sera envoyé dans la Corbeille.

PIXEL reconnaît quelques raccourcis clavier identiques à ceux applicables sous Windows. Vous pouvez sélectionner tous les fichiers et dossiers avec Ctrl + A, vous copiez avec Ctrl + C, coupez avec Ctrl + X et collez avec Ctrl + V. Sachez que Ctrl + C a un autre usage lorsque vous êtes sur la ligne de commande de Linux (voir Chapitre 5) ou depuis le langage Python (Chapitre 11) : ce raccourci permet alors d'annuler l'opération en cours. Dans ce cas, le raccourci pour Copier n'est pas applicable.

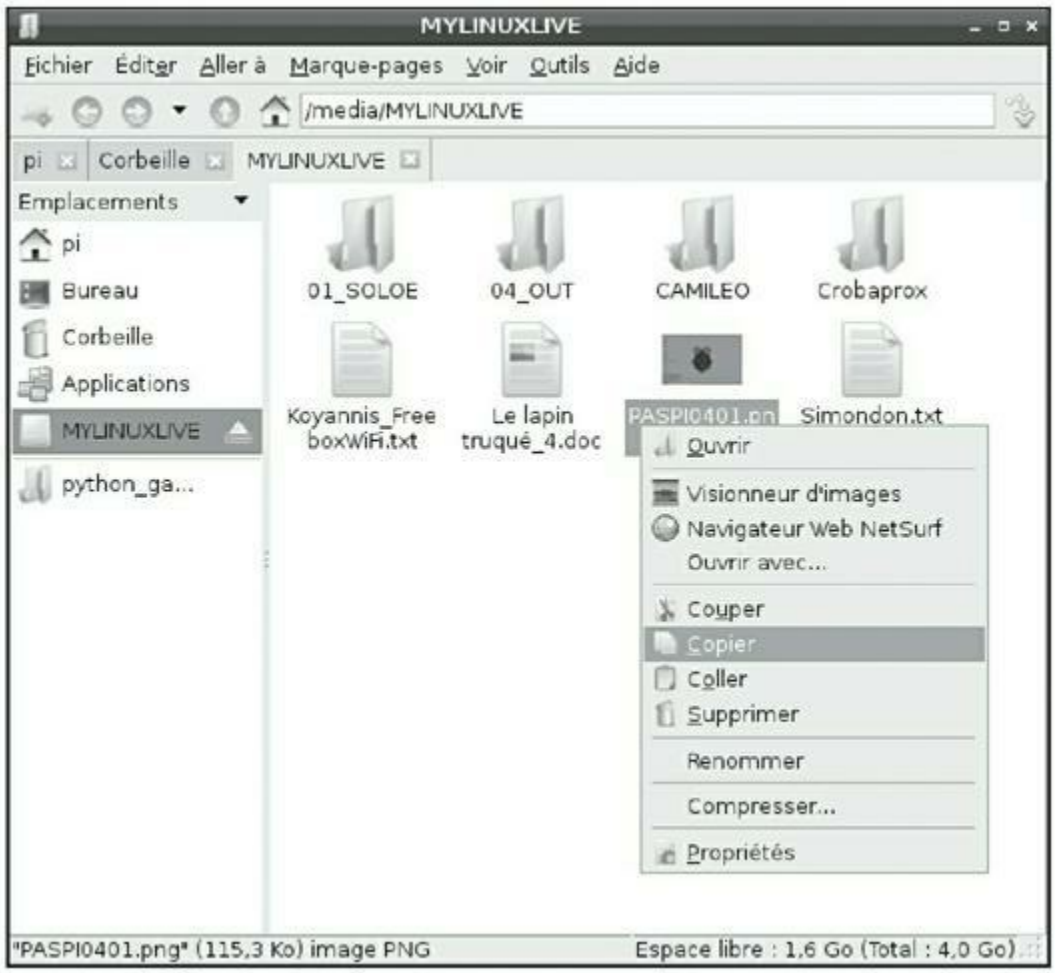


Figure 4.6 : Clic-droit dans un nom de fichier pour accéder au menu local.

Lorsque vous avez besoin de sélectionner quasiment tout le contenu d'un dossier, il est plus efficace de commencer par tout sélectionner avec Ctrl + A, puis de désélectionner les seuls fichiers que vous ne désirez pas en maintenant enfoncée la touche Ctrl tout en cliquant un à un sur chaque fichier à ne pas sélectionner. Vous disposez même d'une commande pour inverser la sélection courante dans le menu **Éditer**.

Création d'un fichier vide ou d'un dossier

Les dossiers sont indispensables pour stocker des fichiers de façon organisée. Vous pouvez ainsi facilement savoir quel fichier se trouve à quel endroit, retrouver facilement un fichier que vous cherchez et copier toute une série de fichiers d'un dossier vers un périphérique amovible.

La création d'un dossier est très facile. Placez-vous d'abord dans le dossier qui doit devenir le dossier parent. Au départ, c'est votre dossier personnel **pi** ou l'un des sous-dossiers qui existent déjà, comme Desktop. Dans le panneau de droite, cliquez droit dans un endroit libre (en dehors d'une icône ou d'un nom) et choisissez **Créer un nouveau** puis choisissez **Dossier** dans le sous-menu. Il ne reste plus qu'à saisir le nom et à cliquer OK pour confirmer. Si vous changez d'avis, utilisez le bouton **Annuler**.

Nous verrons dans le Chapitre 5 qu'il est conseillé de s'astreindre à un minimum de discipline pour choisir les noms des fichiers et des dossiers. Il est notamment conseillé de remplacer les espaces par les caractères de soulignement afin qu'il n'y ait pas d'espace dans les noms. Vous verrez pourquoi dans le prochain chapitre.

La même commande est accessible dans le menu **Fichier** de la barre de menus du Gestionnaire de fichiers.

Vous pouvez également créer un fichier vide au départ. Vous pouvez ainsi vous entraîner à créer des sous-dossiers et des fichiers temporaires, qui vous serviront pour faire des essais de copie et de suppression dans la suite du chapitre sans risquer d'effacer quelque chose d'important.

Suppression de fichiers et de

dossiers

Pour supprimer un fichier ou un dossier, il suffit de cliquer droit dans son nom ou son icône puis de choisir **Envoyer vers la corbeille** dans le menu local. Comme pour la copie, vous pouvez maintenir la touche Ctrl tout en cliquant pour sélectionner plusieurs éléments. Vous pouvez même sélectionner plusieurs objets par zonage en cliquant et maintenant dans une zone libre du panneau droit puis en tirant pour capturer les éléments. Vous pouvez aussi supprimer un élément au moyen de la touche Suppr.

La Corbeille contient les éléments que vous avez supprimés. Elle est directement disponible sous forme d'icône sur le bureau (Figure 4.1). En double-cliquant, vous voyez ce qu'elle contient et pouvez récupérer les éléments avant qu'ils disparaissent définitivement. Par clic-droit, vous pouvez choisir de vider la corbeille.

Lorsque vous décidez de récupérer un élément que vous aviez placé dans la corbeille par mégarde, le fait d'utiliser la commande de restauration replace le fichier dans le dossier dans lequel il se trouvait, ce qui est pratique lorsque vous avez oublié où il était. Vous pouvez même copier le contenu de la corbeille pour le coller ailleurs.

Mode d'affichage des fichiers

Lorsque vous cliquez droit dans un espace vide du panneau droit du Gestionnaire de fichiers, le menu local qui apparaît comporte une option permettant de choisir le mode d'affichage et l'ordre de tri du contenu. Vous pouvez faire trier les noms de fichiers par le radical du nom (avant le suffixe), par heure de modification, par taille ou par type de fichier, dans l'ordre direct ou inverse.

Les différents formats d'affichage du Gestionnaire de fichiers permettent de trouver le format le plus approprié entre un affichage facile à lire de loin et un affichage offrant de la place pour de nombreux noms. Tout se passe dans le menu **Voir**. Par défaut, le Gestionnaire de fichiers utilise l'affichage **Vue en icônes** qui permet de voir un certain nombre de fichiers tout en donnant un indice sur le type de chaque fichier. Le format **Vue en aperçu** est particulièrement utile pour les dossiers contenant des images, puisqu'il permet d'afficher des vignettes miniatures du contenu réel des fichiers. Pour obtenir une liste montrant le plus de noms de fichiers possibles, choisissez le format **Vue compacte** qui présente les noms des fichiers et des sous-dossiers dans des colonnes avec une petite icône à côté du nom.

Enfin, la **Vue en liste détaillée** (Figure 4.7) est celle qui donne le plus d'informations techniques au sujet de chaque fichier, avec un descriptif, la taille et la date de dernière modification. Vous pouvez retrier cette liste en cliquant dans l'en-tête de la colonne désirée. La colonne *Description* permet de regrouper les fichiers du même type. Vous inversez l'ordre de tri en cliquant une seconde fois dans le même en-tête.

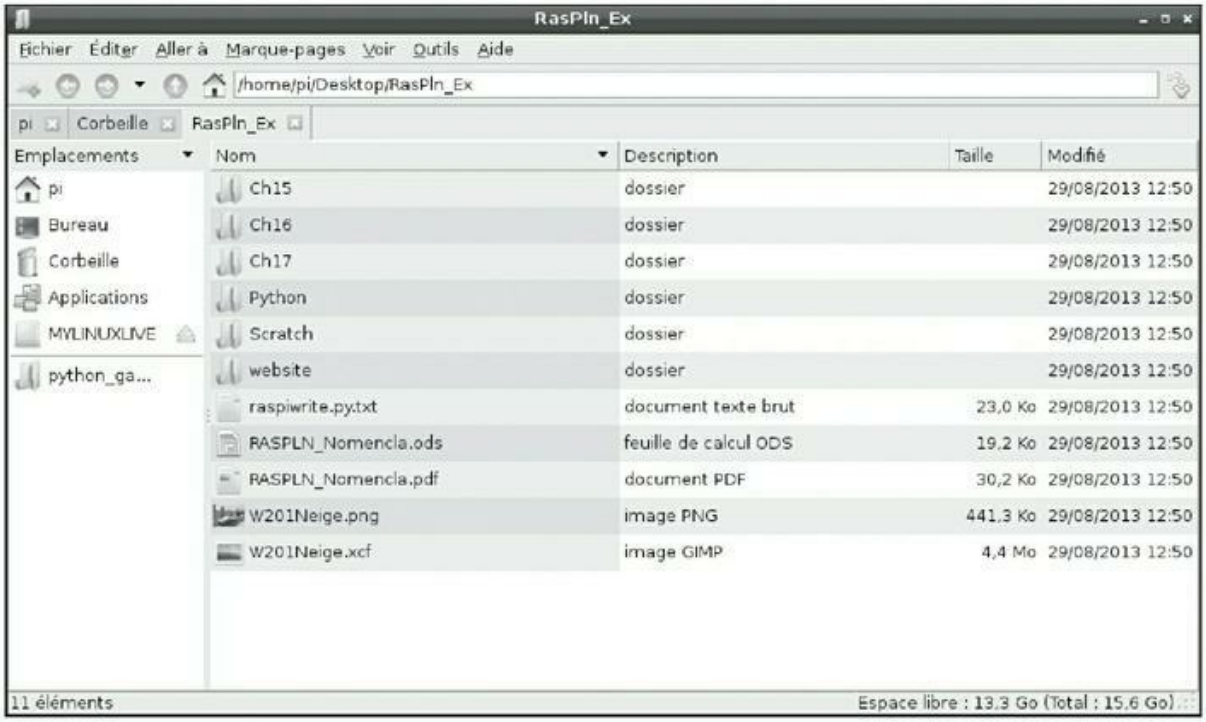


Figure 4.7 : Vue en liste détaillée dans le Gestionnaire de fichiers.

Vous aurez parfois besoin de forcer la mise à jour de l'affichage dans le Gestionnaire de fichiers. Utilisez le raccourci clavier F5 ou la commande du menu **Voir** nommée **Actualiser**.

Le menu **Voir** offre d'autres commandes, notamment au sujet du panneau latéral. Vous pouvez ainsi basculer entre l'affichage de l'arborescence ou accéder à des emplacements favoris. Nous reparlons de la structure arborescente des dossiers et répertoires dans le Chapitre 5. Il vous suffit pour l'instant de savoir qu'il existe une option pour afficher cette structure dans le Gestionnaire de fichiers.

Ouvrir un dossier en root ou dans le terminal

Linux offre un mécanisme très rigoureux qui gère les droits d'accès (permissions) à chacun des fichiers selon le profil de l'utilisateur, et ce en lecture, en modification et en exécution. L'avantage est qu'il devient

assez difficile de faire involontairement des dégâts dans le système d'exploitation du Raspberry Pi. Vous pouvez explorer tous les fichiers utilisés par votre système avec le Gestionnaire de fichiers, mais si vous tentez d'effacer un fichier indispensable, il vous sera rappelé que vous n'en avez pas la permission (voir Figure 4.8).

Pour explorer votre système, revenez à votre dossier principal *pi*, puis cliquez deux fois le bouton pour remonter au dossier parent (revoir Figure 4.4) et observez les noms des dossiers qui s'affichent. Nous présentons les détails des principaux dossiers dans le Chapitre 5.

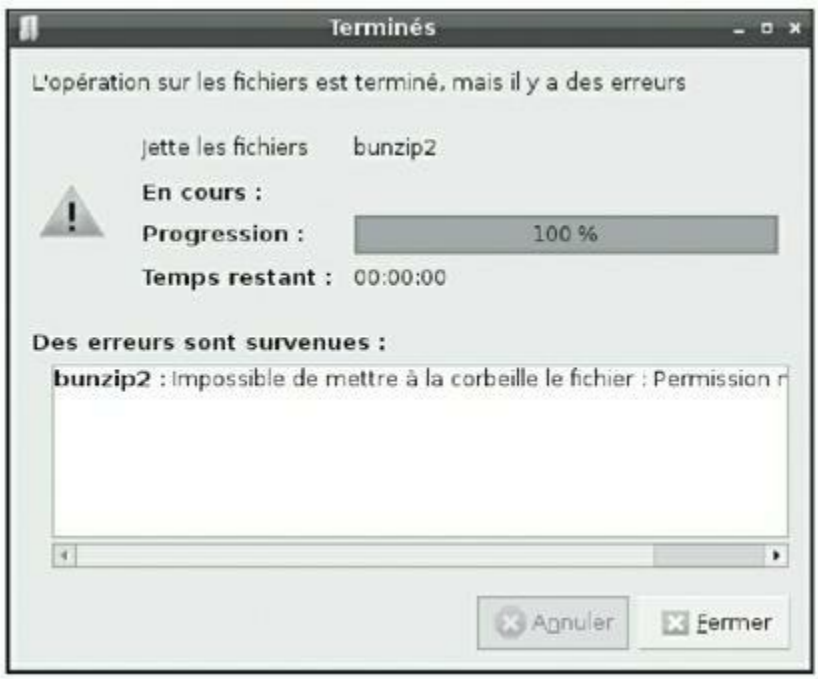


Figure 4.8 : Pardon ! Refus d'autorisation d'effacer un fichier du dossier /boot du Raspberry Pi.

Le menu **Outils** offre une autre option pour ouvrir le répertoire courant dans un terminal, ce qui vous permet d'utiliser toutes les commandes Linux présentées dans le Chapitre 5. C'est souvent la méthode la plus rapide pour effectuer une action précise, mais cela suppose que vous maîtrisiez les commandes en mode texte de Linux. Notez que vous pouvez ouvrir facilement une fenêtre de terminal depuis le Gestionnaire de fichiers au moyen du raccourci clavier F4.

Naviguer sur le Web

En ce qui concerne la navigation Web depuis votre Raspberry Pi, vous avez l'embarras du choix : il est livré avec quatre navigateurs. Celui qui est préconisé, et qui est directement disponible par l'icône de globe en haut à gauche se nomme Chromium. Les trois autres navigateurs

disponibles peuvent être démarrés en saisissant le nom dans la boîte d'exécution du menu général :

- **Dillo :** Ce navigateur est très rapide, mais le rendu des pages Web est modifié parce que les instructions de mise en page sophistiquée et le code JavaScript ne sont pas reconnus. Nous l'avons testé avec plusieurs sites Web, et l'affichage était sur une seule colonne parce que Dillo n'a pas reconnu l'instruction d'en-tête, les barres latérales, le contenu principal et le pied de page. Vous pouvez même désactiver l'affichage des images dans le menu Outils, ce qui accélère encore la lecture des pages chargées. Le navigateur peut tout à fait convenir si vous accédez surtout à des pages de texte ou si votre connexion Web est vraiment lente. En revanche, vous ne pourrez pas profiter des efforts déployés par les concepteurs des sites Web pour leur donner un bel aspect et une meilleure facilité d'utilisation.

- **Netsurf :** Ce navigateur est un peu plus capable que Dillo, mais il ne gère toujours pas le langage JavaScript pour les animations. La plupart des pages Web auront le même aspect que sur un navigateur de PC ou de Mac, mais celles qui ont besoin de JavaScript ne fonctionneront pas (y compris Facebook). Cela dit, Netsurf offre une expérience de navigation plus agréable que Dillo dans la plupart des cas.

- **Epiphany :** C'est le navigateur qui était recommandé avant l'arrivée de Chromium. Il a bien sûr été optimisé pour le Raspberry Pi, et sait

> faire le décodage matériel des vidéos. Il supporte le JavaScript et offre une expérience de navigation Web assez riche. Cela dit, les sites tels que Facebook et Google Maps s'afficheront sans doute un peu plus lentement que d'habitude.

À l'heure actuelle, Flash n'est utilisable que sur les cartes Raspberry Pi 2 et Pi 3. La technologie Flash est encore très utilisée pour les jeux vidéo. Vous ne pourrez pas en profiter si vous avez une carte plus ancienne.

Le navigateur Web Chromium

La Figure 4.9 montre l'aspect général de la fenêtre du navigateur Chromium. L'aspect est très proche de celui des navigateurs que vous connaissez déjà, avec une petite barre d'outils en haut et l'essentiel de la surface de fenêtre dédié à l'affichage de la page Web. Pour lancer le programme, vous utilisez le menu des programmes ou bien vous double-cliquez dans son icône sur le bureau.

Figure 4.9 : Le navigateur Web Chromium.

Lorsque vous connaissez l'adresse du site que vous voulez visiter, vous pouvez directement la saisir dans la barre d'adresse (voir Figure 4.9). Dès que vous commencez à saisir une adresse, un menu historique vous

propose les adresses des pages déjà visitées. Vous pouvez sélectionner directement une des lignes ou continuer à saisir. En fin de saisie, n'oubliez pas de valider en frappant la touche Entrée ou en cliquant dans la flèche vers le bas à droite de la boîte de saisie.

Vous faites défiler le contenu de la page dans le sens vertical avec l'ascenseur sur le bord droit de fenêtre, et vous pouvez utiliser la molette si votre souris en est dotée.

Lorsque vous amenez le pointeur de souris au-dessus d'un lien, le pointeur prend l'aspect d'une main dans un gant. Vous pouvez alors cliquer avec le bouton gauche pour passer à la page visée. Le navigateur mémorise les adresses de toutes les pages visitées dans un historique, ce qui permet d'utiliser les boutons **Page précédente** (voir en Figure 4.9) et **Page suivante**.

Lorsque vous êtes face à une page Web qui comporte des mises à jour régulières, vous pouvez utiliser le bouton **Actualiser** pour forcer la relecture de la page dans son état le plus récent.

Chromium est doté d'un bloqueur de publicité. Vous pouvez le désactiver en accédant aux paramètres correspondants grâce au bouton légendé dans la Figure 4.9.

Recherche dans une page Web

Pour trouver un mot ou une phrase dans une page Web, utilisez la combinaison Ctrl + F une fois que la page est chargée. Vous voyez apparaître la barre de recherche en haut d'écran. Saisissez votre terme. La première occurrence est mise en surbrillance orange. Pour passer à la suivante, utilisez la touche Entrée ou le bouton de la barre de recherche. Pour refermer la barre, frappez Echap ou le bouton de fermeture X sur le bord droit de la barre de recherche.

Navigation par onglets

Comme la plupart des navigateurs actuels, Chromium propose un système d'onglets pour afficher plusieurs pages Web dans la même fenêtre de navigateur. Il suffit de cliquer dans un onglet vide (Figure 4.9) pour ouvrir une nouvelle page qui propose d'afficher l'une des pages les plus visitées. Vous pouvez accepter l'une des propositions ou saisir l'adresse désirée dans la barre d'adresse.

Il suffit de cliquer dans un onglet pour changer l'affichage. Plusieurs pages sont visibles dans la Figure 4.9. Pour refermer un onglet, il suffit de

cliquer le bouton de fermeture à droite de son nom.

Si vous maintenez enfoncée la touche Ctrl tout en cliquant dans un lien, la page correspondante va s'ouvrir dans un nouvel onglet.

Favoris et autres marque-pages

Les favoris ou marque-pages sont indispensables pour mémoriser les adresses des pages dans lesquelles vous voulez revenir. Pour ajouter un favori, utilisez le raccourci Ctrl + D ou cliquez l'étoile dans la barre d'adresse.

La fenêtre pour ajouter un favori ressemble à celle de la Figure 4.10. Par défaut, c'est le titre de la page qui sert de nom au favori, mais vous pouvez bien sûr le modifier. Vous pouvez créer des dossiers pour classer vos favoris grâce à la liste correspondante et stocker un favori dans le dossier spécial de la barre des favoris, affichée sous la barre d'adresse. Vous avez accès très rapidement à son contenu, car cette barre est visible en permanence si vous demandez son affichage. Cliquez le bouton en haut à droite pour accéder au menu. Entrez dans la catégorie **Favoris** puis choisissez **Afficher la barre de favoris**. Le raccourci clavier est Ctrl + Maj + B.

Pour confirmer la création d'un favori, n'oubliez pas de cliquer le bouton **Terminer** ou **Finish** (Figure 4.10).

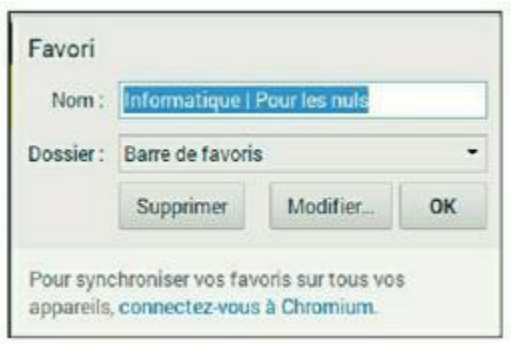


Figure 4.10 : La boîte d'ajout d'un favori de Chromium.

Pour accéder à la liste de vos favoris, ouvrez le menu général (les trois points verticaux en haut à droite) et choisissez **Favoris**.

Vous pouvez gérer vos favoris en accédant au gestionnaire avec Ctrl + Maj + O. Amenez le pointeur au-dessus d'un favori puis cliquez le bouton du menu à droite pour accéder aux options d'édition et de suppression.

Si vous ouvrez une session Google depuis Chromium, vous pourrez profiter de la synchronisation des favoris parmi plusieurs appareils.

Une des fonctions très pratiques de Chromium est la possibilité de mémoriser toutes les pages Web actuellement ouvertes sous forme d'un paquet de favoris dans un seul dossier, ce qui est très pratique lorsque vous effectuez une recherche parmi plusieurs sites. Pour en profiter, ouvrez le menu de Chromium, accédez aux favoris puis choisissez **Ajouter les pages ouvertes aux favoris** (*Bookmark open pages*). Saisissez un nom pour le dossier puis cliquez le bouton de sauvegarde.

Protection de la vie privée

Vous savez que les navigateurs stockent l'historique de toutes les pages visitées. Lorsque vous voulez vous promener sans être surveillé, par exemple pour chercher vos cadeaux de Noël sans que le reste de la famille les devine, il suffit d'ouvrir une fenêtre de navigation en mode privé. Ouvrez le menu général en haut à droite et choisissez **Ouvrir une fenêtre (navigation privée** ou *New private browsing window*). Ce mode spécial est confirmé par la mention correspondante (dans la barre de titre du navigateur). Pour sortir de ce mode, il suffit de fermer la fenêtre concernée.

Vous pouvez également effacer tout l'historique de navigation au moyen d'une commande dans les paramètres. Ouvrez le menu à droite, entrez dans **Paramètres** et descendez parmi les options pour cliquer **Paramètres avancés**. Cliquez enfin **Effacer les données de navigation**.

L'historique permet à d'autres de savoir ce que vous avez visité. Les cookies servent à vous identifier quand vous revenez sur le même site Web, ce qui risque de donner des indices sur les cadeaux que vous avez prévus si quelqu'un d'autre utilise le navigateur pour aller sur le même site. Le pire qui puisse vous arriver est qu'il vous faudra sans doute fournir à nouveau vos données d'identification pour accéder à tous les sites Web concernés.

La messagerie Claws Mail

Claws Mail est un programme open source de messagerie électronique qui est déjà installé. Vous le trouverez dans la catégorie Internet du menu général.

Pour en profiter, vous devez connaître les paramètres d'accès à votre messagerie et notamment votre identifiant et votre mot de passe.

Lors du premier démarrage de Claws Mail, un assistant vous propose de réaliser ce paramétrage. Sachez que vous pouvez modifier les paramètres plus tard, supprimer un compte ou en ajouter un grâce au menu de configuration. Notez que l'option de configuration automatique n'a pas fonctionné lors de nos essais. Préparez-vous donc à saisir les données à la main.

Une fois la configuration achevée, utilisez le bouton en haut à gauche pour relever les messages en attente. L'outil ressemble beaucoup à d'autres clients de messagerie, tels que Thunderbird ou Outlook. Les dossiers sont sur la gauche et les messages sur la droite. Un panneau permet de visualiser le contenu du message sélectionné. Vous pouvez également double-cliquer dans le titre d'un message pour le consulter.

La barre de menus en haut propose de rédiger un nouveau message, de répondre à un message existant, de répondre à tous ou de transférer le message. Vous disposez aussi d'une corbeille.

Le Visionneur d'images

PIXEL fournit un outil très simple pour visionner les photos numériques et autres images. Dans le sous-menu des accessoires, vous trouvez le nom **Visionneur d'images** (*Image Viewer*). En général, vous ne démarrez pas le programme de cette manière, mais par double-clic d'un fichier d'un des types associés à ce programme, c'est-à-dire notamment les photographies et autres fichiers d'images se trouvant dans l'un de vos dossiers.

L'outil affiche l'image en présentant dans le bas de fenêtre une barre d'outils (Figure 4.11). Passons en revue les commandes et options de cette barre d'outils, de gauche à droite :

Figure 4.11 : Le Visionneur d'images.

» **Précédent** : Revient à la précédente image visionnée. Les modifications non enregistrées sont perdues. Au clavier, vous pouvez utiliser la flèche gauche.

» **Suivant** : Va à la photo suivante dans le même dossier. Comme pour le bouton Précédent, les modifications non enregistrées sont perdues. Au clavier, utilisez la flèche droite.

» **Démarrer le diaporama** : Cette commande lance un diaporama de toutes les images trouvées dans le dossier courant. Au départ, la pause entre deux images est de cinq secondes, mais vous pouvez changer cela dans les préférences. Le raccourci clavier correspondant est la lettre **W**.

- **Zoom arrière** : Permet de réduire l'échelle de l'image. Au clavier, utilisez la touche **–** (moins).
- **Zoom avant** : Augmente l'échelle d'affichage. Si l'image devient plus grande que la surface de la fenêtre, des barres de défilement apparaissent pour faire défiler le contenu. Le raccourci clavier est le signe **+**.
- **Ajuster l'image à la taille de fenêtre** : Adapte le taux de zoom à la surface disponible dans la fenêtre. Si l'image est plus petite que la surface, elle ne sera cependant pas agrandie. Le raccourci clavier est la touche **F**. Cette fonction est très utile quand vous avez trop zoomé dans un sens ou dans l'autre.
- **Taille originale** : Restaure le taux de zoom normal pour afficher l'image dans sa taille réelle. Si l'image devient plus grande que la surface disponible, des barres de défilement apparaissent. Au clavier, utilisez la touche **G**.
- **Plein écran** : Occupe toute la surface de l'écran, en masquant les boutons de commande. Dans ce mode, cliquez droit dans l'image pour accéder au menu local des commandes. Pour revenir au mode fenêtré, choisissez **Plein écran** dans le menu local. Au clavier, utilisez la touche **F11** pour basculer entre les deux modes.
- **Pivoter vers la gauche** : Fait tourner l'image de 90 degrés dans le sens antihoraire. Raccourci clavier **L**.

- **Pivoter vers la droite** : Idem dans le sens horaire. Raccourci clavier **R**.
- **Pivoter horizontalement** : Applique un effet de miroir horizontal. Raccourci clavier **H**.
- **Pivoter verticalement** : Applique un effet de miroir vertical. Raccourci clavier **V**.
- **Ouvrir un fichier** : L'icône représentant un dossier permet de charger une nouvelle image. Vous pouvez faire glisser une image depuis un dossier du Gestionnaire de fichiers directement sur l'outil. Cela ne déplace pas le fichier, mais charge son contenu pour affichage.
- **Enregistrer le fichier** : Enregistre les modifications appliquées au fichier (rotation et miroir) en remplaçant l'image de départ. Un avertissement vous en prévient. Le raccourci clavier est **S**.
- **Enregistrer le fichier sous** : Cette commande permet d'enregistrer l'image sous un autre nom pour ne pas écraser la version d'origine.
- **Supprimer le fichier** : Utilisez cette icône ou la touche Suppr pour supprimer l'image affichée. Notez qu'elle n'est pas stockée dans la Corbeille, mais supprimée définitivement. Un avertissement vous le confirme, avant qu'il ne soit trop tard !
- **Préférences** : Cette commande réunit toutes les options du Visionneur d'images permettant de le personnaliser. Vous pouvez ainsi supprimer l'affichage des avertissements avant écrasement ou suppression d'une image, demander un

enregistrement automatique des images retouchées, modifier la couleur de fond et régler la pause du diaporama. Une option permet de faire tourner une image en modifiant la valeur d'orientation dans les données EXIF du fichier (qui sont stockées avec l'image parmi d'autres données comme le lieu de prise de vue). Normalement, vous pouvez laisser l'option active, mais il faut savoir que vous pouvez la désactiver ici.

» **Quitter** : L'icône la plus à droite dans le Visionneur permet de quitter le programme. Méfiez-vous cependant, car la même icône permet de quitter cet outil, mais aussi de quitter la session graphique PIXEL, l'icône étant au même endroit. Cela dit, PIXEL vous demande de confirmer avant de quitter le mode graphique.

L'éditeur de texte

L'éditeur de texte fait partie du sous-menu **Accessoires** du menu général des programmes (voir Figure 4.12). Vous pouvez vous en servir pour écrire du texte simple, mais il ne permet pas de produire un document offrant un bel aspect. Il servira surtout à éditer des fichiers de configuration destinés à l'ordinateur, ou des fichiers de code source tels que ceux des pages Web.

La structure des menus est très logique. Si vous avez utilisé un autre éditeur de texte, vous trouverez rapidement vos marques dans Leafpad.

Le menu **Fichier** sert classiquement à charger un document ou à en créer un nouveau, ainsi qu'à enregistrer et imprimer un fichier. C'est ici que se trouve la commande **Quitter**.

Le menu **Édition** permet d'annuler et de refaire une action, de couper, copier, coller et supprimer ainsi que de sélectionner la totalité du texte. Leafpad reste conforme au niveau des raccourcis à la convention

Windows : Ctrl + C pour copier, Ctrl + V pour coller, Ctrl + X pour couper et Ctrl + A pour tout sélectionner.

Figure 4.12 : L'éditeur de texte.

Le menu **Rechercher** permet de trouver un mot ou une phrase, d'aller directement à une certaine ligne ou de remplacer un mot ou une phrase par autre chose. Une option permet de tout remplacer d'un coup, mais vous pouvez aussi progresser étape par étape. La fonction de recherche/remplacement surligne toutes les occurrences en jaune et l'occurrence active en bleu. Le menu local permet d'avancer dans les deux sens dans la liste des occurrences.

Le menu **Options** (ouvert dans la Figure 4.12) permet de changer la police par l'option **Fonte** (mais le nombre de polices installées au départ est bien plus limité que ce dont vous avez l'habitude). Vous activez l'option de retour à la ligne automatique, ce qui évite de voir apparaître une barre sous la fenêtre. L'option **Numéro de lignes** permet d'ajouter des numéros dans la marge gauche. Enfin, l'option **Indentation automatique** est fort pratique dans les programmes : dès que vous validez par Entrée, elle demande à l'outil de reproduire le nombre d'espaces en début de ligne qui ont été insérées dans la ligne précédente.

Personnaliser le bureau graphique

Différentes options sont offertes pour adapter l'environnement PIXEL à vos préférences et le rendre plus simple à utiliser. Comme dans tous les environnements graphiques fenêtrés, vous pouvez changer l'apparence et le comportement. Pour accéder à ces options, ouvrez le menu général puis la section **Préférences** et choisissez **Appearance settings** (ou **Préférences du Bureau**).

Dans les options relatives au bureau, vous pouvez changer l'image de fond, les couleurs du bureau (si vous voulez un fond uni) et la couleur des légendes des icônes. Des options sont disponibles pour donner accès directement à votre dossier Documents et aux disques amovibles sur le bureau. La page de la barre de menus permet de changer la taille, la position, la couleur de la barre qui est normalement en haut d'écran. La page Système permet de changer la police utilisée dans tout l'environnement et les couleurs des barres de titre des fenêtres.

Vous pouvez aussi accéder à ces réglages par clic-droit sur le bureau puis choix des préférences de bureau dans le menu local.

Un autre module de préférences nommé **Clavier et Souris** permet de paramétrer le clavier et la souris, par exemple d'inverser l'action des deux boutons pour les gauchers.

Ajouter et enlever des applications

Vous pouvez rechercher puis installer des applications en mode texte depuis la ligne de commande comme décrit dans le Chapitre 5, mais l'approche par le bureau est beaucoup plus confortable. Dans le menu des applications, accédez aux préférences puis choisissez **Add/Remove software** (Ajouter/supprimer des logiciels). Il faut absolument que vous disposiez d'une connexion Internet.

La Figure 4.13 montre l'aspect général de l'outil. Tout en haut à gauche, vous disposez d'une zone de saisie pour vos recherches. Saisissez le début du nom du programme que vous cherchez, ou une phrase du style **puzzle games**. Le volet de gauche propose des catégories, permettant de limiter les résultats en fonction du thème désiré.

La surface principale de la fenêtre contient la liste des paquetages correspondant à ce que vous avez demandé. Utilisez l'ascenseur sur le bord droit pour naviguer dans la liste. Les logiciels qui comportent une coche et sont affichés en gras sont ceux déjà installés. Cliquez pour obtenir une description. Pour demander l'installation d'un nouveau paquetage, il suffit de cocher sa case. Pour désinstaller un paquetage, vous décochez sa case.

Vous pouvez ainsi sélectionner plusieurs opérations avant de cliquer le bouton OK qui va lancer l'installation et la désinstallation. Vous devrez saisir votre mot de passe, qui est normalement **raspberry**. Sachez que le traitement peut prendre un certain temps. Il est souvent conseillé de procéder à l'installation de plusieurs programmes en même temps, ce qui permet de vaquer à d'autres occupations pendant le téléchargement.

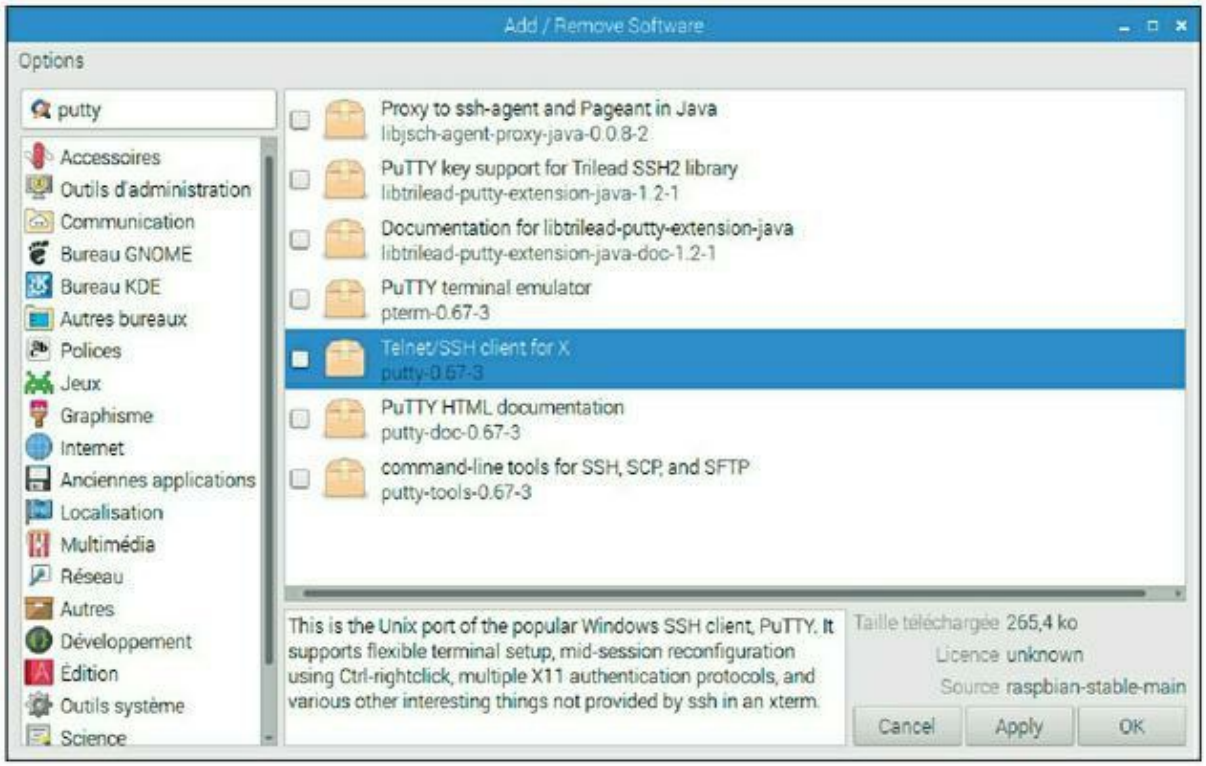


Figure 4.13 : Le programme d'ajout et de suppression de logiciels.

L'outil est organisé pour télécharger automatiquement les logiciels qui sont indispensables au fonctionnement de ceux que vous demandez d'installer, ce que l'on appelle les dépendances. Par exemple, lorsque nous avons installé le jeu *Brain Party*, le logiciel a automatiquement installé le paquetage de données qui doit l'accompagner.

Sauvegarde des données

Vous pouvez facilement sauvegarder vos fichiers en les copiant vers une clé USB au moyen du Gestionnaire de fichiers vu dans ce chapitre, ou grâce à des commandes en mode texte comme le montre le prochain chapitre. Si le nombre de fichiers à sauvegarder est important, et si vous avez effectué de nombreuses opérations de personnalisation du système, il peut être plus intéressant de réaliser une copie entière de la carte système. Une application a été prévue à cet effet ; elle porte le nom **SD Card Copier** et se trouve dans le sous-menu des accessoires.

Pour pouvoir utiliser cet outil, il faut disposer d'un second lecteur de carte SD branché sur un port USB. Vous pouvez aussi demander d'effectuer la duplication vers une clé USB. En revanche, le lecteur de carte SD est indispensable pour restaurer vers une carte microSD qui doit pouvoir servir à démarrer votre Raspberry Pi.

La carte SD ou la clé USB que vous allez utiliser comme cible va voir son contenu totalement effacé. Vérifiez d'abord que vous n'avez plus besoin de ce qu'elle contient.

La boîte de dialogue de l'outil est visible dans la Figure 4.14. Elle propose deux listes déroulantes permettant de choisir la source et la cible de l'opération. Si vous n'êtes pas sûr du nom de la carte SD qui contient le système, ouvrez la deuxième liste, **Copy to device**. Le nom de la carte système ne peut pas y apparaître, car il est impossible de copier vers la carte système.

Figure 4.14 : L'outil SD Card Copier.

Cet outil sait réaliser une copie image de NOOBS et de Raspbian, mais il se peut qu'il ne fonctionne pas avec d'autres systèmes d'exploitation ou d'autres formats de carte mémoire.

Prenez patience, car l'opération peut prendre un certain temps sans afficher d'information de progression.

Déconnexion et arrêt système

Lorsque vous n'avez plus besoin d'utiliser votre machine, vous devez effectuer un arrêt avant de couper l'alimentation. Les commandes pour arrêter ou redémarrer votre RasPi sont disponibles dans le sous-menu **Shutdown** du menu général. Faites bien la différence avec la commande de déconnexion **Logout** qui sert à refermer votre compte utilisateur, sans arrêter le système. Dans ce cas, vous vous reconnectez au moyen du nom d'utilisateur **pi** et du mot de passe **raspberry**.

Une fois que l’arrêt est terminé, attendez que la petite LED verte arrête de clignoter. Seule la rouge doit briller. À ce moment, vous pouvez couper l’alimentation.

Chapitre 5

L'interpréteur Linux shell

DANS CE CHAPITRE :

- **Découvrir le système de fichiers Linux**
- **Créer, supprimer et naviguer dans les fichiers et dossiers (répertoires)**
- **Trouver et installer des logiciels**
- **Gérer les comptes utilisateurs du Raspberry Pi**
- **Personnaliser l'interpréteur avec vos propres commandes**

L'interpréteur est le programme essentiel de tout ordinateur pour entrer en communication avec lui au moyen de commandes tapées au clavier. Depuis l'environnement graphique du bureau, vous y accédez dans une fenêtre ne pouvant afficher que des lettres et des chiffres, mais aucun graphique, une fenêtre de terminal. L'interpréteur du Raspberry Pi se nomme *bash*, et c'est une version qui est utilisée dans de nombreuses autres distributions Linux. Précisons que bash est une abréviation de *Bourne Again Shell*, ce qui rappelle qu'il s'agit d'un remplaçant de l'ancien interpréteur Bourne.

Vous allez apprendre au cours de ce chapitre comment tirer profit de l'interpréteur pour gérer votre Raspberry Pi. De nombreux arguments militent en faveur d'un apprentissage de cet interpréteur. C'est notamment la technique la plus efficace pour réaliser certaines tâches, bien plus efficace que de passer par le bureau graphique. De plus, il est toujours utile d'apprendre à maîtriser l'interpréteur de Linux. C'est un système d'exploitation de plus en plus populaire et le Raspberry Pi vous permet de le découvrir et d'en apprendre les principes. Cela vous permettra aussi de mieux comprendre ce qui se passe dans les coulisses du système de votre Raspberry Pi.

Depuis l'environnement graphique, cliquez en haut l'icône montrant un carré noir avec les deux signes >_. C'est celle du **Terminal**. Vous démarrez ainsi une session d'interpréteur dans une fenêtre en mode texte. Le Terminal est aussi accessible dans la catégorie Accessoires du menu général.

Ne vous inquiétez pas si l'écran se vide lorsque vous utilisez l'interpréteur. Vous pouvez sortir du mode veille en frappant n'importe quelle touche du clavier. Prenez l'habitude d'utiliser une des flèches du curseur.

L'invite de commande

Lorsque vous avez réussi à vous connecter au système du Raspberry Pi, vous voyez apparaître un message qui correspond à son **invite** (*prompt*). Vous voyez clignoter le curseur, ce qui signifie qu'il est prêt à recevoir la saisie d'une commande de votre part :

```
pi@raspberrypi ~ $
```

Au premier regard, cette invite peut sembler inutilement complexe (pourquoi ne pas afficher simplement « OK » ou « Prêt » ?). En fait, elle réunit plusieurs informations précieuses. Analysons son contenu pièce par pièce :

- `pi` : Cela rappelle le nom de l'utilisateur qui a ouvert la session. Nous verrons plus loin dans ce chapitre comment créer d'autres comptes utilisateurs sur le Raspberry Pi. Le nom d'utilisateur sera différent si vous ouvrez une session avec un autre compte.
- `raspberrypi` : C'est le nom de la machine hôte, celui que vont voir les autres ordinateurs du même réseau local pour identifier cette machine.
- `~` : Sous Linux, on utilise le terme *répertoire* au lieu de *dossier*, mais ce sont des synonymes. Cette partie de l'invite vous permet de savoir quel est le répertoire courant. Ce symbole ~ (tilde, celui que

l'on trouve sur les n espagnols) est un raccourci qui symbolise votre répertoire personnel ou répertoire d'accueil (*home*). Son affichage dans l'invite confirme que vous êtes actuellement dans votre racine personnelle. Nous avons vu dans le Chapitre 4 que c'est dans ce répertoire que vous devez stocker vos fichiers et créer vos sous-dossiers. D'ailleurs, un utilisateur normal n'a aucun droit pour stocker des fichiers ailleurs que dans son répertoire personnel ou dans les sous-dossiers de ce répertoire.

» `$` : Le signe dollar rappelle que vous êtes un utilisateur normal et non un superutilisateur. Si vous étiez superutilisateur, vous verriez le symbole `#` à la place. Nous verrons plus loin dans ce chapitre ce qu'il en est des droits d'accès des utilisateurs.

Exploration du système Linux

Vous pouvez sans risque passer en revue les fichiers et répertoires de votre carte mémoire SD. Puisque vous êtes un utilisateur normal, il vous sera interdit de supprimer ou de déplacer les fichiers système. Vous pouvez donc explorer tout le contenu de la carte SD sans avoir peur de modifier ou de supprimer quelque chose d'important.

Affichage des fichiers et répertoires

La commande qui permet d'afficher la liste des fichiers et des répertoires s'écrit `ls` (*listing*). Puisque vous démarrez dans votre répertoire personnel, vous pouvez saisir cette commande pour voir s'afficher les noms des répertoires et des fichiers qu'il contient. Voici un exemple d'affichage. Précisons que nous utilisons dans les exemples du texte en gras pour singulariser ce que vous devez saisir. Ce qui est affiché en réponse par l'ordinateur est en texte non gras.

```
pi@raspberrypi ~ $ ls
Desktop    Downloads   Pictures   python_games
Videos
Documents  Music       Public     Templates
```

Linux est sensible à la casse, ce qui signifie qu'il distingue les majuscules des minuscules. Les commandes LS, ls, Ls, et lS sont quatre instructions différentes. Sous Linux, un S majuscule et un s minuscule ne sont pas la même chose, de la même manière que pour nous un a et un z sont deux lettres différentes. Si vous vous trompez dans la casse des lettres, la commande ne sera pas reconnue dans l'interpréteur. Si vous oubliez une majuscule dans un nom de fichier ou en ajoutez une, Linux va vous répondre que le fichier n'existe pas. Lorsque nous utiliserons des commandes plus complexes et des options un peu plus loin, vous verrez que certaines commandes font bien la distinction entre majuscules et minuscules pour désigner des options différentes.

Changement de répertoire

Au départ, le contenu du répertoire personnel indique des noms affichés en bleu, ce qui rappelle que ce sont des répertoires et non des fichiers. Nous pouvons donc entrer dans ces répertoires pour voir leur contenu. La commande qui permet de changer de répertoire s'écrit `cd`, et vous la faites suivre du nom du répertoire dans lequel vous désirez vous rendre, comme ceci :

```
pi@raspberrypi ~ $ cd python_games
```

Vous remarquez que l'invite du système change pour confirmer que vous avez changé de répertoire courant. Vous pouvez vérifier que vous avez bien changé de répertoire courant en demandant l'affichage de son contenu avec la commande `ls`.

Contrôler les types de fichiers

Vous pouvez en savoir plus au sujet d'un fichier au moyen de la commande `file`. Après le nom de la commande et une espace, indiquez le nom du fichier désiré. Vous pouvez faire afficher les données de plusieurs fichiers dans la même commande en séparant les noms par une espace :

```
pi@raspberrypi ~/python_games $ file boy.png
match0.wav wormy.py
boy.png: PNG image data, 50 x 85, 8-bit/color
RGBA, non-interlaced
match0.wav: RIFF (little-endian) data, WAVE
audio, Microsoft PCM, 16 bit,
mono 44100 Hz
wormy.py: Python script, ASCII text executable
```

Cette commande `file` donne un certain nombre de détails. Non seulement vous apprenez le type de données que contient chaque fichier (dans l'exemple, une image, un fichier audio et un fichier de code source Python), mais vous découvrez aussi la taille de l'image et le fait que l'audio est en mono.

Si vous êtes un utilisateur chevronné, vous aurez sans doute deviné le type de fichier d'après les extensions de noms (les mentions .png, .wav, et .py à la fin des noms de fichiers). Sous Linux, il n'est pas obligatoire que les fichiers portent des extensions, et c'est pourquoi la commande `file` peut se révéler très utile. (En pratique, la grande majorité des applications utilisent des extensions de noms de fichiers et les utilisateurs trouvent cela pratique, car cela permet de deviner les contenus des fichiers par simple lecture du nom.)

La commande `file` peut être appliquée à un répertoire. Lorsque vous êtes dans votre répertoire personnel `pi`, vous pouvez vérifier la nature des répertoires `Desktop` et `python_games` de la manière suivante :

```
pi@raspberrypi ~ $ file Desktop python_games
Desktop: directory
python_games: directory
```

La belle affaire ! Nous venons de nous voir confirmer qu'il s'agit de répertoires. Il peut sembler illogique d'utiliser une commande portant le nom anglais « fichier » (*file*) pour en apprendre plus au sujet d'un répertoire, mais cela permet de souligner un concept fondamental de Linux : dans ce système, tout est fichier, même un disque dur et une connexion réseau. Il s'agit dans tous les cas de fichiers, dans le sens Linux.

Retour au répertoire de niveau

supérieur

Nous savons déjà utiliser la commande `cd` pour descendre dans un sous-répertoire du répertoire de départ. Pour revenir en arrière dans le répertoire parent du répertoire courant, il faut connaître une technique particulière. Le répertoire `python_games` est un sous-répertoire du répertoire `pi`. Autrement dit, `pi` est le répertoire parent.

Pour revenir au répertoire parent, vous faites suivre la commande `cd` du symbole spécial `..` (deux points consécutifs). Si vous utilisez cette commande dans le sous-répertoire python_games, vous revenez à votre répertoire personnel, ce que confirme l'affichage de l'invite tilde.

```
pi@raspberrypi ~/python_games $ cd ..
pi@raspberrypi ~ $
```

Rappelons que le symbole ~ représente le répertoire personnel. Le nom complet de ce répertoire est identique à votre nom d'utilisateur, ce qui est donc `pi` pour le premier utilisateur. Le répertoire parent de ce répertoire personnel porte le nom `home`, car c'est le répertoire contenant tous les répertoires personnels de tous les comptes d'utilisateurs.

Lorsque vous êtes dans votre répertoire personnel, essayez d'utiliser la commande `cd ..` pour remonter d'un niveau dans le répertoire qui se nomme `home`. Si vous relancez la même commande, vous vous trouvez dans la racine de tous les répertoires du système, c'est-à-dire dans le répertoire dont le nom se limite à la seule barre oblique et que l'on appelle *root* (racine). Essayez maintenant de remonter jusqu'au répertoire racine puis d'afficher son contenu :

```
pi@raspberrypi ~ $ cd ..
pi@raspberrypi /home $ cd ..
pi@raspberrypi / $ ls
bin boot dev etc home lib lost+found media mnt
opt proc root run sbin selinux srv sys
tmp usr var
```

Vous pouvez naviguer dans les différents sous-répertoires depuis la racine avec la commande `cd`. Servez-vous de `ls` pour voir le contenu de chacun, de `cd` pour passer à un autre répertoire et de `file` pour voir la nature des fichiers dans chaque répertoire.

L’arborescence de répertoires

Lorsqu’il s’agit de présenter la manière dont sont organisés les répertoires sur un ordinateur, on utilise souvent la métaphore de l’arbre. Un arbre possède un tronc dont partent de nombreuses branches, chacune portant des branches secondaires et ainsi de suite, jusqu’à atteindre les feuilles.

Votre Raspberry Pi ne possède qu’un seul répertoire racine, et tous les répertoires en découlent, avec leurs sous-répertoires en cascade.

La Figure 5.1 schématise les principaux répertoires de l’arborescence du système Linux. Tous les sous-répertoires ne sont pas présentés, ni depuis la racine, ni depuis les sous-répertoires principaux. Le schéma suffit à vous montrer où se situe votre répertoire personnel relativement aux autres. Vous pouvez prendre ce schéma comme une carte. Lorsque vous êtes au niveau de la racine et désirez vous rendre dans le sous-répertoire, vous pouvez voir le chemin qu’il faut emprunter pour entrer dans votre répertoire personnel en passant par `home`.

Lorsque vous êtes au niveau de la racine, vous voyez une vingtaine de répertoires. Ils contiennent tous les fichiers de données, tous les fichiers des programmes et tous les fichiers système de votre Raspberry Pi. Vous pouvez tout à fait entrer dans chacun des répertoires et jeter un œil en utilisant `file` pour connaître la nature des fichiers.

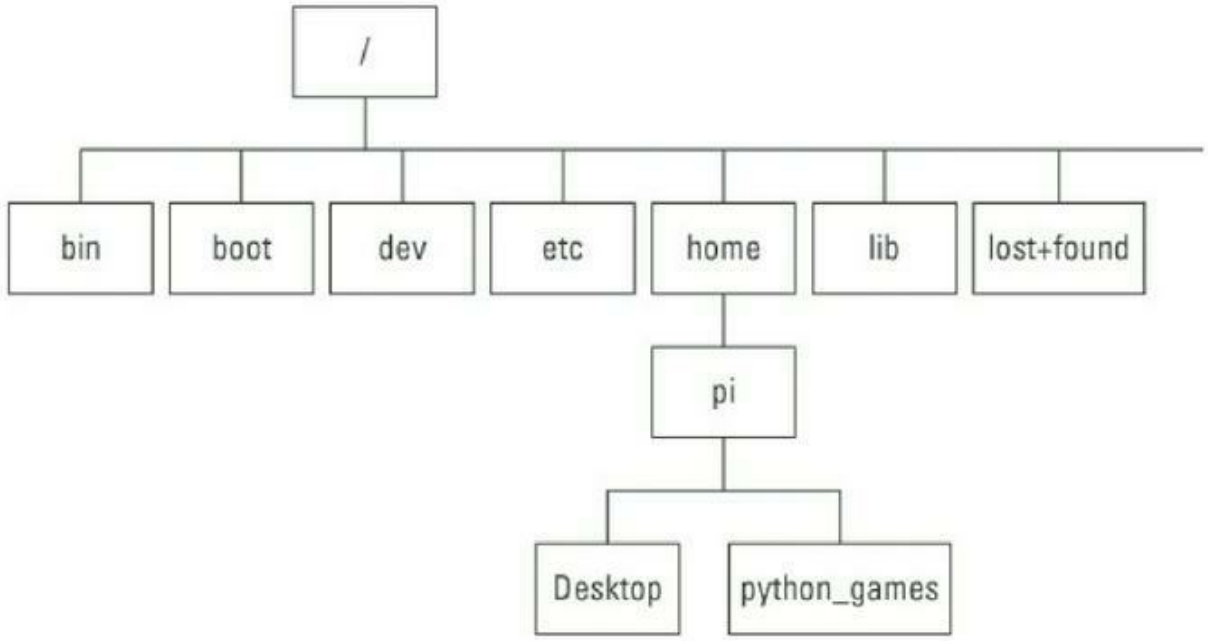


Figure 5.1 : Structure arborescente partielle de votre Raspberry Pi.

Vous aurez très rarement besoin d’accéder à la plupart de ces répertoires, mais il est toujours utile de savoir à quoi servent les principaux d’entre eux :

- `bin` : Abréviation de *binaries*, ce répertoire réunit de petits programmes qui sont utilisés comme des commandes dans l'interpréteur, et notamment `ls` et `mkdir`, dont nous nous servirons plus loin pour créer un répertoire.
- `boot` : C'est ici que se trouve le noyau Linux, qui est le cœur du système d'exploitation, ainsi que quelques fichiers de configuration contenant des paramètres techniques pour le Raspberry Pi. L'Annexe A montre comment modifier le fichier *config.txt* qui se trouve ici afin de régler quelques paramètres de la machine.
- `dev` : Ce répertoire contient une liste des périphériques (des disques durs et des connexions réseau) que gère le système.
- `etc` : Ce répertoire accueille de nombreux fichiers de configuration qui sont applicables à tous les utilisateurs.
- `home` : Nous avons déjà présenté ce répertoire qui sert de base pour tous les sous-répertoires personnels des différents comptes utilisateurs. C'est le seul endroit dans lequel un utilisateur est autorisé à créer ou modifier des fichiers (par défaut).
- `lib` : Ce répertoire contient des librairies de fonctions ou programmes partagés qui sont nécessaires à différents programmes du système d'exploitation.
- `lost+found` : Sous ce nom étrange se trouve un répertoire dont vous n'aurez normalement

jamais besoin. Il est utilisé lorsque le système de fichiers est altéré et cherche à se restaurer en partie.

- `media` : C'est dans ce répertoire que sont stockés les détails d'un périphérique amovible – tel qu'une clé USB – lorsque vous le connectez au système et qu'il est automatiquement détecté dans l'environnement de bureau graphique.
- `mnt` : Sont stockés ici les détails des périphériques amovibles que vous montez (connectez) manuellement au système (nous indiquons la procédure dans l'Annexe A).
- `opt` : Ce répertoire recueille les logiciels optionnels du Raspberry Pi. En général, il sert sous Linux aux logiciels que vous installez vous-même. Sur le Raspberry Pi, les programmes sont généralement installés dans `/usr/bin`.
- `proc` : Ce répertoire très spécial est utilisé par le noyau du système Linux pour vous permettre d'accéder à des informations concernant le fonctionnement du système. Pour exploiter ces informations, vous devez posséder des connaissances spécialisées, mais il n'est pas inutile de jeter un œil. Essayez d'entrer la commande `less /proc/cpuinfo` et vous verrez un affichage concernant le processeur du Raspberry Pi. La commande `less / proc/meminfo` permet de voir la quantité de mémoire du Raspberry Pi et le taux d'occupation. (Nous verrons plus loin comment utiliser la commande `less`, mais il suffit pour l'instant de savoir que la

touche **Q** permet de sortir de l'outil.) En revanche, si vous utilisez la commande `file` dans ce répertoire, les fichiers sembleront vides, ce qui montre bien que c'est un répertoire spécial, dont le contenu est mis à jour en permanence.

- `root` : Un utilisateur normal n'a absolument aucun droit d'effectuer des modifications dans ce répertoire, car il est utilisé uniquement par le superutilisateur *root*, seul compte utilisateur autorisé à réaliser toutes les opérations possibles sur la machine. Raspberry Pi vous invite à ne pas utiliser ce compte spécial, mais à utiliser plutôt la commande préfixe `sudo` lorsqu'il s'agit d'émettre une commande avec le même niveau de droits que le superutilisateur. Nous verrons plus loin dans ce chapitre comment nous servir de ce préfixe pour installer des logiciels.
- `run` : Ce répertoire est apparu récemment dans Linux. Il constitue un lieu de stockage des données pour les programmes en cours d'exécution, en leur garantissant que ces données seront disponibles au prochain démarrage du système. En effet, les données du répertoire nommé `tmp` peuvent à tout moment être effacées par les outils de nettoyage disque. Par ailleurs, le répertoire nommé `usr` n'est pas toujours accessible au démarrage sur tous les systèmes Linux (il peut être situé sur un autre sous-système de fichiers).
- `sbin` : Ce répertoire contient des logiciels qui sont normalement réservés au superutilisateur.

- `selinux` : Ce répertoire est destiné au mécanisme de sécurité de Linux nommé Security-Enhanced Linux. Il est vide au départ.
- `srv` : Cet autre répertoire vide est parfois utilisé par Linux pour stocker les répertoires de données des services de téléchargements tels que le FTP qui permet de transférer des fichiers via Internet.
- `sys` : Ce répertoire est réservé aux fichiers du système d'exploitation Linux.
- `tmp` : Ce fichier reçoit des fichiers temporaires, qui peuvent être effacés à tout moment.
- `usr` : Ce répertoire reçoit les fichiers des programmes et fichiers associés qui sont utilisables et exécutables par les utilisateurs normaux.
- `var` : Ce dernier répertoire rassemble les fichiers dont la taille peut varier, comme c'est le cas des bases de données et des fichiers de journalisation. Vous pouvez afficher le contenu du journal des messages système au moyen de la commande `less /var/log/messages` (utilisez la flèche Bas pour défiler et la touche Q pour quitter).

Chemins d'accès relatifs et absolus

Nous avons déjà appris comment se déplacer entre un sous-répertoire et un répertoire parent dans la structure arborescente. Dans un environnement graphique, vous cliquez dans un nom de dossier pour l'ouvrir, et cliquez à nouveau pour entrer dans un sous-dossier, et ainsi de suite. C'est très simple et c'est pourquoi cela est très populaire, mais dans une structure complexe, avancer ainsi pas à pas peut lasser.

Lorsque vous savez où vous voulez aller, l'interpréteur permet de vous rendre directement dans un sous-répertoire en une seule commande, en spécifiant un *chemin d'accès*, c'est-à-dire une description de tous les différents répertoires menant à celui désiré. Il existe des chemins d'accès relatifs et des chemins d'accès absolus. Un *chemin d'accès relatif* dépend du répertoire courant (vous remontez d'un répertoire, vous descendez d'un répertoire et vous y êtes !). En revanche, un *chemin d'accès absolu* correspond à une adresse physique : ce chemin reste valable où que vous soyez lorsque vous émettez la commande.

Un chemin d'accès absolu commence par la racine du système de fichiers, qui est symbolisé par la barre oblique isolée. Après ce symbole viennent les noms des sous-répertoires qui mènent à celui désiré. Le chemin d'accès absolu vers votre répertoire personnel `pi` s'écrit `/home/pi`. Où que vous soyez, vous revenez directement à cet endroit de la manière suivante :

```
cd /home/pi
```

Pour entrer directement dans le sous-répertoire `Desktop`, vous écrivez :

```
cd /home/pi/Desktop
```

Pour revenir à la racine en un geste, vous écrivez simplement :

```
cd /
```

Vous pouvez également utiliser le symbole de votre répertoire personnel qui est le signe tilde pour revenir directement dans ce répertoire :

```
cd ~
```

Pour saisir le symbole tilde, utilisez la touche **AltGr** avec la touche du **2** du clavier principal.

Vous pouvez vous servir de ce symbole comme point de départ d'une sorte de chemin d'accès absolu local, afin d'aller dans un de vos sous-répertoires :

```
cd ~/Desktop
```

Un chemin d'accès relatif exprime un déplacement par rapport à votre point de départ. Rappelons que le répertoire courant est toujours affiché

dans l'invite de commande, mais vous pouvez aussi le faire afficher au moyen de la commande suivante :

```
pwd
```

Notez que la commande `pwd` n'affiche pas le symbole de votre répertoire personnel (`~`), mais l'expression complète sous forme de chemin absolu `/home/pi`.

Dans un chemin relatif qui fait entrer dans un sous-répertoire sont mentionnés les noms des sous-répertoires successifs séparés par une barre oblique. Si vous revoyez la Figure 5.1, vous pouvez deviner un répertoire nommé `home` qui contient le répertoire `pi`, lui-même contenant un répertoire `Desktop`. Lorsque vous êtes dans le répertoire principal `home`, vous pouvez entrer directement dans `Desktop` en indiquant le chemin relatif `pi/Desktop`, comme ceci :

```
pi@raspberrypi /home $ cd pi/Desktop
pi@raspberrypi ~/Desktop $
```

Cette technique permet d'accéder à n'importe quel sous-répertoire. Vous pouvez également utiliser un chemin relatif pour remonter dans la structure au moyen du symbole spécial `..` (qui correspond au répertoire parent). Toujours en nous servant de la Figure 5.1, il est possible de remonter depuis le sous-répertoire `Desktop` puis de redescendre dans le sous-répertoire `python_games` en passant par le parent commun à ces deux répertoires, comme ceci :

```
pi@raspberrypi ~/Desktop $ cd ../python_games
pi@raspberrypi ~/python_games $
```

La commande précédente demande de remonter d'un niveau, ce qui fait revenir dans `pi`, puis de descendre dans le sous-répertoire `python_games`. Vous pouvez utiliser le symbole double point plusieurs fois pour remonter de plusieurs niveaux. Voici par exemple comment aller de votre répertoire `pi` vers le répertoire système `boot` :

```
pi@raspberrypi ~ $ cd ../../boot
pi@raspberrypi /boot $
```

Cette commande vous fait remonter d'abord dans le répertoire `home`, puis dans le répertoire racine du système puis vous redescendez dans le répertoire `boot`.

Vous choisirez entre un chemin absolu et un chemin relatif en fonction du contexte. Lorsque votre cible est assez proche de votre répertoire courant, le chemin relatif est le plus pratique. Dans les autres cas, le chemin absolu est plus efficace. Ce genre de chemin ne sert pas qu'à changer de répertoire courant. Vous pouvez également vous en servir en ajoutant un nom de fichier au bout du chemin pour appliquer une action à ce fichier. Voici comment vous pourriez utiliser la commande `file` :

```
file /boot/config.txt
```

Vous allez découvrir de nombreuses autres commandes dans ce chapitre qui s'appliquent à des fichiers. Vous pourrez utiliser votre connaissance des chemins d'accès pour désigner un fichier qui ne se trouve pas dans votre répertoire courant.

Distinguez bien les répertoires absolus et relatifs. Méfiez-vous notamment de votre utilisation de la barre oblique isolée `/`. Votre chemin d'accès doit commencer par une barre oblique seulement lorsque vous voulez spécifier un chemin d'accès absolu qui commence par le répertoire racine.

Un autre symbole permet d'aller visiter un sous-répertoire puis de revenir rapidement au répertoire précédent :

```
cd -
```

Lorsque vous émettez cette commande, l'interpréteur affiche le nom du répertoire précédent puis en fait le répertoire courant.

Enfin, pour revenir directement à votre répertoire personnel, utilisez la commande `cd` sans aucun argument, comme ceci :

```
pi@raspberrypi /boot $ cd
pi@raspberrypi ~ $
```

Options de listage avancées

Vous pouvez utiliser la commande `ls` pour afficher le contenu d'un autre répertoire en spécifiant le chemin d'accès :

```
pi@raspberrypi ~ $ ls /boot
```

Dans cette commande, vous ne quittez pas votre répertoire personnel, mais vous obtenez l'affichage du contenu du sous-répertoire `/boot`.

Lorsque vous fournissez une information à la suite d'un nom de commande, comme un chemin d'accès ou un nom de fichier, cela s'appelle un *argument de commande*. Quasiment toutes les commandes Linux acceptent des arguments qui déterminent la cible de leur action (c'est par exemple le cas de `cd` et de `file`). Certaines commandes acceptent aussi des options qui ne sont pas la même chose. Une option détermine la manière dont la commande doit travailler. Toutes les options commencent par un tiret suivi d'une ou plusieurs lettres caractéristiques.

La commande `ls` possède plusieurs options qui sont rassemblées dans le Tableau 5.1. Vous pouvez par exemple revenir dans votre répertoire personnel pour saisir ceci :

```
pi@raspberrypi ~ $ ls -R
```

Vous obtenez l'affichage du contenu du répertoire courant, mais également du contenu de tous ses sous-répertoires (`Desktop` et `python_games`). Le R correspond à la Récursivité.

Lorsque vous utilisez des options et des arguments, les options doivent être indiquées d'abord. Voici le format générique d'une commande Linux :

```
commande  -options  arguments
```

Voyons en guise d'exemple comment utiliser l'option `-X` pour obtenir l'affichage du contenu du sous-répertoire `python_games`. Cette option fait regrouper les fichiers par type (.png, .py, et .wav). Voici la commande à émettre :

```
pi@raspberrypi ~ $ ls –X ~/python_games
```

Vous pouvez spécifier plusieurs options en même temps en les rassemblant derrière le tiret. Pour afficher le contenu de tous les sous-répertoires (option **R**), tout en regroupant les résultats par type de fichiers (option **X**) et en utilisant les symboles pour afficher les noms des répertoires (option **F**), la commande devient celle-ci :

```
pi@raspberrypi ~ $ ls -RXF
```

La Figure 5.2 donne un exemple de résultats affichés. Vous pouvez voir qu'un point isolé est utilisé pour désigner le répertoire courant. Le chemin d'accès à la première ligne de résultats est donc ce point isolé. Ce symbole point unique utilise le même principe que le symbole couple de points qui symbolise le répertoire parent.

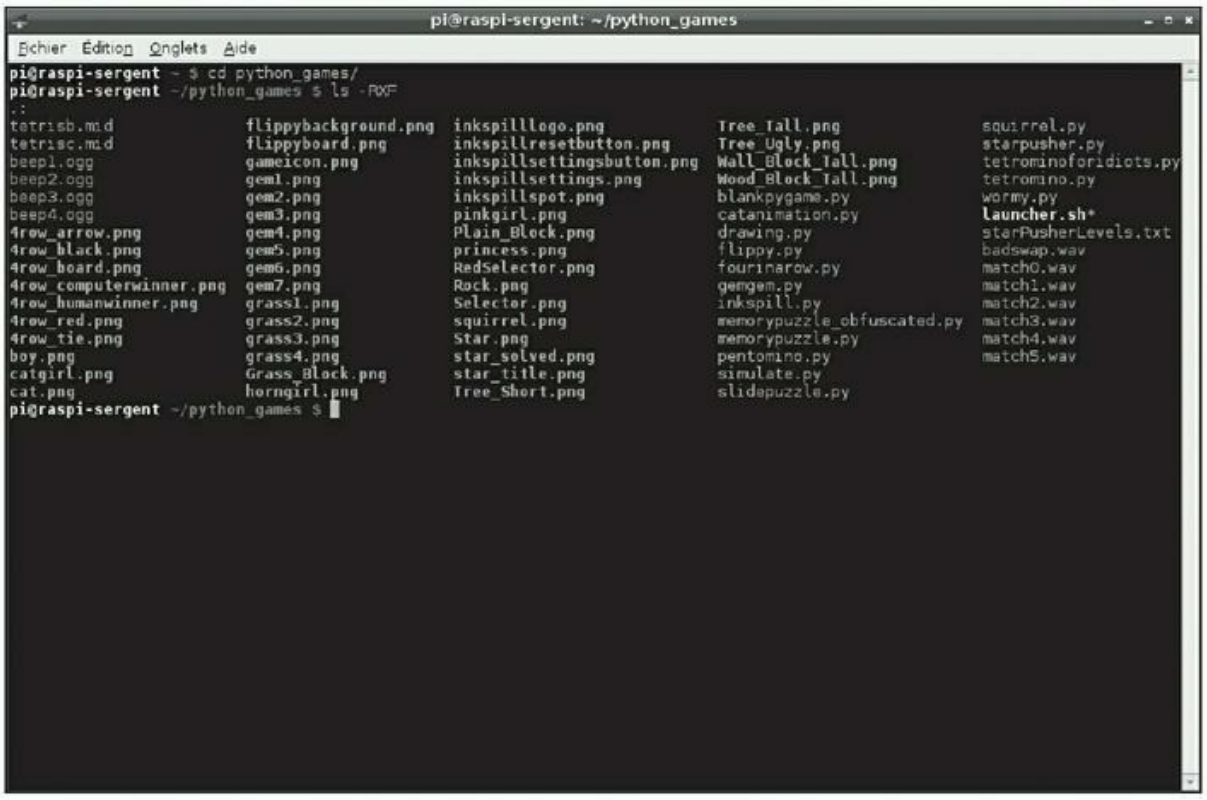


Figure 5.2 : Liste récursive avec les sous-répertoires triés par types de fichiers et des symboles pour les dossiers et exécutables.

Lorsque vous faites des essais avec la commande `ls` ou n'importe quelle autre commande, profitez de la commande `clear` pour effacer l'écran de temps à autre.

Tableau 5.1 : Options de la commande ls.

Option	Description
-1	(C'est le chiffre 1, pas la lettre l) Affiche les résultats sur une seule colonne et non par lignes.
-a	Affiche tous les fichiers y compris les fichiers cachés, ces derniers commençant par un signe point. Les fichiers cachés sont créés par le système et sont nécessaires. Vous pouvez

	créer des fichiers cachés en faisant commencer le nom par le signe point.
-F	Ajoute un symbole de type à côté du nom de fichier. Les répertoires comportent une barre oblique après le nom et les fichiers exécutables un astérisque.
-h	Affiche dans le format long les tailles de fichiers en kilooctets, mégaoctets et gigaoctets pour éviter de devoir faire les calculs. Le « h » signifie humain.
-l	Affiche les résultats au format long avec tous les droits d'accès aux fichiers, la date de dernière modification et la taille. Le « l » minuscule signifie long.
-m	Affiche les résultats sous forme d'une liste séparée par des virgules (ce qui peut servir pour un autre programme).
-R	Récursivité. Affiche les fichiers et répertoires du répertoire courant et tous les contenus des sous-répertoires. Vous pouvez par exemple afficher la liste gigantesque de tous les fichiers de votre Raspberry avec la commande `ls -R` émise depuis la racine. Soyez patient ; pour abandonner, utilisez Ctrl + C.
-r	Option d'inversion de l'affichage. Par défaut, les résultats sont présentés dans l'ordre alphabétique, et cette option inverse l'ordre. Remarquez au passage que `-r` et `-R` n'ont aucun rapport.
-S	Trie les résultats par taille.
-t	Trie les résultats selon la date et l'heure de dernière modification (time).
-X	Trie les résultats selon l'extension des noms de fichiers.

Droits d’accès ou permissions

Une des options indispensables de la commande `ls` est celle du format long, `-l`. En voici un exemple :

```
pi@raspberrypi ~/ $ ls -l
total 40
-rw-r--r-- 1 pi pi 256   Nov 18 13:53
mesnotes.txt
drwxr-xr-x 2 pi pi 4096  Oct 28 22:54 Desktop
drwxrwxr-x 2 pi pi 4096  Nov 17 13:40
python_games
drwxr-xr-x 2 pi pi 4096  Nov 3  17:43 boulot
-rw-r--r-- 1 pi pi 20855 Nov 12 2010
spacegame.sb
```

Le résultat peut vous sembler étrange, mais la lecture devient plus simple lorsque l’on travaille ligne par ligne et de droite à gauche. Chaque ligne correspond à une entité (un fichier ou un répertoire). Le nom est indiqué sur la droite et la date et heure de dernière modification juste à gauche du nom. Lorsque la date est assez ancienne, c’est l’année qui est affichée à la place de l’heure, comme vous pouvez le voir pour le dernier fichier de la liste.

Le chiffre à gauche du nom de mois correspond à la taille du fichier. Trois des éléments de cet exemple sont des sous-répertoires (`Desktop`, `python_games` et `boulot`). Ils ont une taille unique de 4 096 octets, même si leur contenu est très différent. En effet, un répertoire est un fichier au sens Linux. La valeur indiquée ici correspond à l’occupation sur disque ou carte mémoire du répertoire et non au volume des données que contient ce répertoire. Par défaut, les tailles sont exprimées en octets, mais vous disposez de l’option `-h` pour faire afficher les multiples ; 4 096 octets deviendraient par exemple dans ce cas 4K.

Tout ce qui se trouve à gauche de la taille concerne les *droits d’accès*, également appelés *permissions*. Ces droits stipulent qui a le droit d’utiliser chaque fichier et dans quelles conditions. Le système Linux a été conçu dès le départ pour permettre une utilisation sécurisée par plusieurs utilisateurs en même temps. Les droits d’accès constituent donc un concept fondamental de Linux.

Nombreux sont ceux qui pensent pouvoir utiliser leur Raspberry Pi sans se soucier des droits d'accès, mais ils vous en apprennent beaucoup sur le fonctionnement de ce système, et vous aurez intérêt à savoir comment fonctionnent ces droits d'accès au cas où vous auriez envie d'aller un peu plus loin dans la maîtrise du système.

Les droits d'accès sur un fichier sont divisés en trois catégories : les droits du *propriétaire* (*owner*) du fichier (c'est normalement la personne qui a créé le fichier), les droits du *groupe* (dont le propriétaire fait partie, mais ce n'est pas le seul) et les droits de tous les *autres* utilisateurs (on parle également de droits pour le « monde »).

Dans l'affichage au format long, nous voyons que le mot **pi** est présenté deux fois pour chaque fichier. Les deux colonnes correspondent au nom du propriétaire du fichier ou du répertoire (c'est la colonne la plus à gauche) et au nom du groupe qui possède le fichier. Ici, le nom est le même dans les deux cas parce que le système crée dès le départ un groupe portant le même nom que le compte utilisateur. En fait, le groupe pourrait porter le nom `etudiant` afin de permettre la gestion regroupée de tous les étudiants sur la même machine.

Les dix lettres de la première colonne de gauche constituent un masque codé des types de droits pour l'élément concerné pour chacune des trois catégories. Pour décoder ce masque, il faut le découper en quatre morceaux, comme le montre le Tableau 5.2. Dans ce tableau, nous avons analysé le masque de droits d'accès des deux fichiers de l'exemple.

Tableau 5.2 Analyse des droits d'accès.

Type de fichiers	Propriétaire	Groupe	Monde
-	rw-	r--	r--

Les deux types d'éléments que vous rencontrerez le plus souvent sont les fichiers et les répertoires. Pour un fichier, le premier caractère du masque est le symbole tiret alors qu'un répertoire comporte un `d` (*directory*). Ces deux situations sont visibles dans l'exemple.

Viennent ensuite les trois trios de droits d'accès pour le propriétaire, son groupe et le reste du monde. Voici les trois types de droits possibles dans chaque catégorie :

» Droits en **lecture** (*read*) : Permet d'ouvrir et d'afficher le contenu d'un fichier ou de lister le contenu d'un répertoire.

- Droits en **écriture** (*write*) : Permet de modifier le contenu d'un fichier ou de créer ou de supprimer des fichiers dans un répertoire.
- Droits en **exécution** (*execute*) : Permet de considérer un fichier comme un programme et d'en lancer l'exécution ou d'entrer dans un répertoire au moyen de la commande `cd`.

Tout cela semble logique, mais il y a deux pièges : tout d'abord, vous ne pouvez lire ou écrire dans un répertoire que si vous avez également les droits en exécution pour ce répertoire. Deuxièmement, vous ne pouvez renommer ou supprimer un fichier que si les droits que vous avez sur le répertoire qui le contient vous y autorisent, même si vous avez les droits en écriture pour le fichier.

Les droits sont symbolisés par les lettres **r** (pour la lecture, *read*), **w** (pour l'écriture, *write*) et **x** pour l'exécution. Lorsqu'un droit n'est pas attribué, la lettre est remplacée par un tiret. Vous pouvez donc voir dans le Tableau 5.2 que le propriétaire peut lire et écrire ses fichiers, mais que son groupe et le reste du monde ne peuvent que les lire.

Analysons le masque du dossier `Desktop` de l'exemple qui ce lit `drwxr-xr-x`. La première lettre nous confirme qu'il s'agit d'un répertoire. Le trio de lettres suivant (`rwx`) confirme que le propriétaire peut lire, écrire et exécuter, ce qui correspond à l'affichage du contenu, à l'ajout et à la suppression de fichiers et à l'entrée dans le répertoire pour faire ces opérations. Le trio de caractères suivant (`r-x`) nous indique que le groupe peut entrer dans le répertoire (exécution) et afficher son contenu (lecture), mais qu'il n'est pas autorisé à créer ou supprimer des fichiers. Le dernier trio de caractères (`r-x`) indique que les autres utilisateurs ont les mêmes droits que le groupe.

Vous disposez de certaines commandes pour intervenir sur les droits d'accès : `chmod` pour modifier les droits, `chown` pour modifier le propriétaire d'un fichier et `chgrp` pour modifier le groupe. Nous n'avons pas assez de place dans ce livre pour les décrire en détail, mais la fin du chapitre montre comment accéder à la documentation des commandes. Lorsque vous avez besoin de modifier les droits, il est plus simple d'utiliser l'environnement graphique. Lorsque vous êtes dans le Gestionnaire de fichiers, cliquez droit dans un nom (voir le Chapitre 4) et demandez à accéder à ses propriétés. Passez à la page des permissions

(Figure 5.3) dans laquelle vous pouvez modifier par des listes les droits d'accès du fichier.

Figure 5.3 : Modification des droits d'accès dans le Gestionnaire de fichiers.

Contrôler le défilement avec less

Le problème de la commande d'affichage `ls` c'est qu'elle peut vous noyer dans la montagne d'informations qu'elle présente à l'écran. L'affichage défile trop vite pour être utilisable. Si vous utilisez la fenêtre **LX Terminal** depuis l'environnement graphique, vous disposez d'un ascenseur sur le bord droit pour revenir en arrière sur ce qui a été affiché.

En mode texte, la meilleure solution est d'utiliser la commande nommée `less` qui permet de formater un affichage en plusieurs pages écran.

Pour envoyer les résultats affichés par une commande à la commande `less`, il suffit d'interconnecter les deux commandes avec le caractère barre verticale ou caractère de canalisation :

```
ls ¦ less
```

Vous pouvez ainsi vous déplacer dans l'affichage une ligne à la fois avec les flèches Haut et Bas du curseur ou une page à la fois avec les touches PgUp (ou la touche B) et PgDn (ou l'espace). Vous pouvez même lancer une recherche en frappant la barre oblique puis en saisissant l'élément à chercher et en validant par Entrée. Pour terminer, frappez la touche **Q** en majuscule ou en minuscule.

Vous pouvez à tout moment abandonner l'exécution d'une commande Linux par frappe du raccourci Ctrl + C.

La commande `less` permet aussi d'afficher le contenu d'un fichier texte. Il suffit de donner le nom du fichier en argument :

```
less /boot/config.txt
```

C'est une excellente technique pour savoir ce que contiennent les fichiers pendant que vous explorez Linux. La commande vous prévient si le fichier que vous demandez d'afficher est de type binaire, ce qui rend l'affichage inintelligible. Vous pouvez donc sans risque essayer la commande sur n'importe quel fichier et abandonner lorsque le programme vous avertit. Il faut savoir que l'affichage de code binaire à l'écran peut donner des résultats assez étranges, et même aller jusqu'à altérer le jeu de caractères utilisé par l'interpréteur.

Vous pouvez afficher uniquement les dix premières lignes d'un fichier, par exemple pour connaître sa version en utilisant la commande `head` suivie du nom de fichiers.

Vous disposez maintenant de tous les outils qui vont vous permettre d'explorer votre système Linux !

Optimiser la saisie de commandes

Maintenant que vous connaissez quelques commandes de base, l'heure est venue de vous faire découvrir quelques astuces pour être encore plus efficace dans l'interpréteur.

Apprenez tout d'abord que l'interpréteur mémorise toutes les commandes déjà saisies dans un historique, ce qui vous permet d'économiser de la saisie. Pour utiliser la dernière commande, tapez deux points d'exclamation puis validez par Entrée. Pour une commande plus ancienne, utilisez la flèche Haut du clavier pour faire afficher tour à tour toutes les commandes déjà émises en remontant dans l'historique. La flèche Bas permet de parcourir l'historique dans l'autre sens. Vous pouvez retoucher la commande rappelée avant de valider par la touche Entrée.

L'interpréteur essaye de deviner ce que vous êtes en train de saisir en complétant automatiquement le texte lorsque vous utilisez la touche de Tabulation. Vous pouvez vous en servir pour les noms de commandes comme pour les noms de fichiers. Saisissez par exemple ceci :

```
cd /bo
```

Dès que vous frappez la touche Tab, le seul chemin d'accès possible est inséré, */ boot/*.

Cette technique est extrêmement utile lorsque vous devez saisir des noms de fichiers et des chemins d'accès longs. Lorsque la réponse ne convient pas, c'est que vous n'avez pas saisi assez de caractères. Fournissez quelques lettres de plus et utilisez à nouveau la fonction pour qu'il n'y ait plus d'ambiguïté.

Créer un fichier par redirection (< et <<)

Avant de voir comment on supprime et on copie un fichier, créons quelques fichiers avec lesquels nous pourrons faire des tests sans risque.

Vous pouvez sous Linux envoyer ce qui est normalement affiché à l'écran par une commande vers un fichier, c'est-à-dire *rediriger les données*. Cela permet par exemple de constituer un fichier contenant la liste du contenu d'un répertoire pour l'analyser à tête reposée ou charger la liste dans un éditeur de texte. Pour renvoyer dans un fichier le contenu de l'affichage, vous utilisez l'opérateur `>` suivi d'une espace et du nom de fichier destinataire :

```
ls > listing.txt
```

Vous n'êtes pas forcé de spécifier l'extension `.txt` pour Linux, mais c'est une pratique habituelle qui vous aide à connaître le type du fichier, et vous sera très utile si vous comptez transférer ce genre de fichier vers une machine sous Windows.

Essayez d'utiliser la précédente commande deux fois de suite sans changer de répertoire, mais en désignant deux répertoires différents. Affichez ensuite le contenu du fichier `listing.txt` avec la commande `less`. Vous constatez que Linux peut parfois manquer de garde-fou : lors de la première exécution de la commande, le fichier reçoit le contenu du répertoire concerné. Lors de la seconde exécution, tout ce contenu est supprimé et remplacé par la liste du nouveau répertoire. Linux suppose que vous savez ce que vous faites et ne présente même pas un avertissement en cas d'écrasement d'un contenu de fichier.

Plusieurs commandes sont disponibles pour afficher des informations à l'écran, et ces informations peuvent être envoyées dans un fichier :

- `echo` : Affiche ce que vous venez de saisir à l'écran. La commande permet de résoudre des calculs mathématiques lorsque vous les placez entre deux doubles paires de parenthèses avec un signe dollar en préfixe, comme ceci :

```
echo $((5*5))
```

- `date` : Affiche l'heure et la date courantes.
- `cal` : Affiche le calendrier du mois en cours, avec la date du jour en gras. Pour voir le calendrier de toute l'année, ajoutez l'option **– y**.

Pour ne pas écraser le contenu antérieur d'un fichier texte, mais ajouter des données à la fin, vous utilisez l'opérateur double signe supérieur à >> comme ceci :

```
pi@raspberrypi ~ $ echo Fichier produit le >
fictest.txt
pi@raspberrypi ~ $ date >> fictest.txt
pi@raspberrypi ~ $ cal >> fictest.txt
pi@raspberrypi ~ $ echo $((20+2)) jours avant mon
anniv! >> fictest.txt
pi@raspberrypi ~ $ less fictest.txt
Fichier produit le
Fri Jun 1 14:40:43 GMT 2018
     June 2018
Su Mo Tu We Th Fr Sa
               1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

```
22 jours avant mon anniv !
```

Ce mécanisme de redirection va vous permettre de créer quelques fichiers pour faire des essais de copie et de suppression. Vous pouvez même créer des fichiers vides en redirigeant vers un fichier à partir de rien, comme ceci :

```
> fictest1.txt
```

Conseils de nommage des fichiers sous Linux

Si vous comptez utiliser l'interpréteur de temps à autre, nous vous conseillons d'adopter quelques règles pour créer les fichiers, afin de simplifier votre quotidien sous Linux. Notez que ces conseils s'appliquent aussi pour créer des fichiers depuis l'environnement graphique, mais ils prennent tout leur sens lorsque vous essayez de les manipuler dans l'interpréteur en mode texte.

Voici nos cinq conseils :

- N'utilisez en règle générale que des lettres minuscules, afin de ne pas avoir à vous souvenir où vous avez utilisé des lettres capitales dans un nom de fichier.
- N'utilisez jamais d'espace dans vos noms de fichiers. Cela vous obligerait à les délimiter par des guillemets ou des apostrophes. Sans ces délimiteurs, Linux considère chaque mot suivi d'une espace comme un nom de fichier ou une option différente. Remplacez toutes vos espaces par le caractère de soulignement qui offre quasiment le même aspect visuel.
- Ne faites jamais commencer un nom de fichier par le tiret. Vous risquez de le faire confondre avec une option de commandes.

- N’utilisez nulle part le caractère / qui est utilisé dans les chemins d’accès.
- Évitez tous les caractères suivants dans les noms de fichiers : l’apostrophe ('), le point d’interrogation (?), l’astérisque (*), les guillemets ("), l’antibarre (\), les crochets droits ([]), et les accolades ({}). Lorsque vous êtes forcé d’utiliser un de ces caractères dans un nom de fichier au niveau de l’interpréteur, il faut en neutraliser la signification spéciale en le faisant précéder par le caractère antibarre ou bien délimiter tout le nom de fichier par des guillemets (à supposer qu’il n’y ait pas déjà de guillemets dans le nom).

Créer un répertoire (mkdir)

Vous savez maintenant que la structure arborescente constituée de répertoires et de sous-répertoires constitue une technique universelle pour bien gérer le stockage des fichiers sur un ordinateur. Voyons comment créer un sous-répertoire depuis votre répertoire personnel au moyen de la commande `mkdir` :

```
mkdir montaf
```

Vous pouvez même créer plusieurs sous-répertoires d’un coup :

```
pi@raspberrypi ~ $ mkdir montaf college jeux
pi@raspberrypi ~ $ ls
college jeux montaf
```

La commande `mkdir` n’est pas la seule à permettre de créer plusieurs éléments dans la même ligne de commande. Plusieurs autres commandes acceptent un nombre multiple d’arguments qu’ils vont traiter dans l’ordre d’apparition. Voici comment afficher tour à tour le contenu de deux répertoires différents :

```
ls ~ /boot
```

La commande `mkdir` n'est pas très bavarde pendant son traitement, mais il suffit de spécifier l'option `-v` (verbeux) pour qu'elle affiche un commentaire lors de la création de chaque sous-répertoire. Vous le verrez dans le prochain extrait d'affichage.

Pour créer un répertoire avec un sous-répertoire en même temps, plutôt que de créer le répertoire, d'entrer dans ce répertoire, de créer un sous-répertoire, d'entrer dans le répertoire, *etc*., profitez de l'option `-p` :

```
pi@raspberrypi ~ $ mkdir –vp
montaf/rediger/bouquins
mkdir : created directory 'montaf'
mkdir : created directory 'montaf/rediger'
mkdir : created directory
'montaf/rediger/bouquins'
```

Supprimer un fichier (rm)

Après avoir créé quelques fichiers et sous-répertoires, vous devez disposer maintenant d'un certain nombre d'éléments, et vous pouvez donc faire un peu de ménage.

Pour supprimer un fichier sous Linux, vous utilisez la commande `rm` (*remove*). Soyez très vigilant avec cette commande, car il n'y a pas de corbeille permettant de récupérer un fichier effacé par mégarde. Un utilisateur Linux chevronné pourrait être capable de récupérer un fichier supprimé avec un outil particulier et assez de temps et de patience. Pour un utilisateur normal, non doté d'un tel outil, il faut retenir que lorsque l'on demande de supprimer un fichier à Linux, il le fait vraiment et c'est définitif.

Voici le format générique de la commande `rm` :

```
rm options nomfic
```

Comme la commande `mkdir`, rien n'est affiché pendant le traitement, sauf si vous ajoutez l'option `-v`. Voici par exemple comment supprimer un fichier portant le nom *lettre.txt* :

```
pi@raspberrypi ~ $ rm –v lettre.txt
removed 'lettre.txt'
```

La commande accepte plusieurs arguments, ce qui permet de supprimer plusieurs fichiers en séquence, comme ceci :

```
pi@raspberrypi ~ $ rm –v lettre.txt lettre2.txt
removed 'lettre.txt'
removed 'lettre2.txt'
```

C'est dans ce genre d'utilisation qu'il faut être très vigilant. Supposons que vous ayez créé deux fichiers portant le nom `old index.html` et `index.html`. Le dernier des deux est la nouvelle page d'accueil de votre site Web sur laquelle vous avez sué plusieurs jours. Vous voulez supprimer l'ancienne version. Vous émettez benoîtement la commande suivante :

```
pi@raspberrypi ~ $ rm –v old index.html
rm : cannot remove 'old': No such file or
directory
removed 'index.html'
```

Avez-vous deviné le piège ? À cause de l'espace dans le nom du premier fichier (*old index.html*), la commande `rm` a cru que vous vouliez supprimer deux fichiers, le premier portant le nom `old` et le deuxième portant le nom `index.html`. Elle vous informe qu'elle n'a pas pu trouver le premier fichier nommé `old`, mais n'a aucun problème pour supprimer votre belle dernière version du site Web `index. html`. Dommage !

Pour éviter ce genre de bévue, prenez l'habitude de toujours spécifier l'option `- i` (interactif) qui oblige la commande à vous demander confirmation fichier par fichier pour la suppression. Avec cette option, nous aurions évité cette catastrophe, comme ceci :

```
pi@raspberrypi ~ $ rm -vi old index.html
rm : cannot remove 'old': No such file or
directory
rm : remove regular file 'index.html'?
```

Surtout pas ! Lorsque la question est posée, vous répondez **Y** pour confirmer la suppression ou bien **N** pour conserver le fichier et passer au suivant, le cas échéant.

La principale raison pour laquelle nous vous déconseillons d'utiliser des espaces dans les noms de fichiers est le risque de supprimer un fichier par mégarde. La bonne manière de supprimer un fichier qui contient des espaces consiste à délimiter le nom par des apostrophes :

```
pi@raspberrypi ~ $ rm –vi 'old index.html'
```

Sélection multiple avec les métacaractères * et ?

Il arrive souvent qu'un répertoire contienne toute une série de fichiers possédant un radical identique. Par exemple, l'appareil photo du fils de l'auteur produit des fichiers portant des noms dans le style suivant :

```
img_8474.jpg
img_8475.jpg
img_8476.jpg
img_8477.jpg
img_8478.jpg
```

Pour copier d'un geste ou supprimer toute cette série de fichiers, il n'est pas question de répéter la même commande pour chaque nom de fichier individuellement. Quand on leur explique bien, les ordinateurs sont très doués pour répéter la même commande avec une petite variation. Laissons donc faire l'interpréteur.

La solution consiste à utiliser les *métacaractères* ou caractères génériques. Au lieu de spécifier le nom de fichier exact, vous spécifiez un motif pour désigner tous les fichiers qui commencent par les trois lettres *img* ou tous les fichiers portant l'extension *.jpg*. Le métacaractère `*` (astérisque) remplace de zéro à x caractères. En spécifiant `*.jpg`, vous désignez tous les fichiers se terminant par *.jpg*, quels que soient les caractères situés en début de nom et quelle que soit leur quantité. De même, le métacaractère `?` (point d'interrogation) remplace un seul caractère. En spécifiant `img ? .jpg`, vous désignez *img1.jpg*, *img2.jpg* et *imgb.jpg*, mais pas *img11.jpg* ni aucun autre nom de fichier plus long.

Vous disposez également des crochets droits pour désigner une série limitée de valeurs à une certaine position dans le nom. Pour désigner tous les fichiers qui commencent par une des lettres a, b ou c, vous écrivez

`[abc]*`. Pour ne désigner que les fichiers parmi ceux-là qui se terminent par *.jpg*, vous écrivez `[abc]*. jpg`.

Le Tableau 5.3 constitue une référence abrégée de tous les métacaractères utilisables dans les noms de fichiers et de répertoires, avec des exemples.

Tableau 5.3 : Métacaractères Linux sur Raspberry Pi.

Métacaractère	Signification	Exemple	Résultat de l'exemple
?	Un caractère isolé	`photo ? .jpg`	Tout fichier commençant par photo avec un seul caractère après et avant l'extension .jpg. Exemple : photo1.jpg ou photox.jpg, mais pas photo10.jpg.
*	Plusieurs caractères ou aucun	`*photo*`	Tout fichier contenant le mot photo dans le nom.
[...]	Un des caractères de la plage	`[abc]*`	Tous les fichiers commençant par a, b, ou c.
[^...]	N'importe quel caractère sauf ceux de la plage	`[^abc]*`	Tout fichier qui ne commence pas par la lettre a, b ou c.
[a-z]	Toute lettre unique dans la plage	`[a-c]*.jpg`	Tout fichier commençant par a, b, ou c et se terminant par .jpg.
[0-9]		`photo[2-5].jpg`	

	Tout chiffre unique dans la plage		Désigne photo2.jpg, photo3.jpg, photo 4.jpg, et photo5.jpg.

Vous pouvez utiliser les métacaractères partout où vous utilisez un nom de fichier. Voici par exemple comment supprimer tous vos fichiers commençant par les trois lettres *img* :

```
rm –vi img*
```

Pour supprimer tous les fichiers se terminant par *.txt* :

```
rm –vi *.txt
```

Soyez très attentif avec l'utilisation de l'espace lorsque vous utilisez des métacaractères. Supposez que vous ayez ajouté par mégarde une espace dans l'exemple précédent, comme ceci :

```
rm –vi *  .txt
```

Catastrophe ! L'interpréteur comprend que vous demandez de détruire *****, c'est-à-dire tous les fichiers, puis (s'il est toujours là) un fichier nommé *.txt*. Mais vous avez heureusement utilisé l'option `-i` et vous allez voir la commande vous demander de confirmer pour chaque fichier. Les gens évitent souvent cette option lorsqu'il s'agit de supprimer de nombreux fichiers, car elle les oblige à confirmer individuellement la suppression, ce qui revient à passer presque autant de temps qu'en n'utilisant pas de métacaractère.

Pour plus de sécurité, vous pouvez vérifier quels seront les fichiers concernés par un motif de métacaractère au moyen de la commande `file` à appliquer avant d'utiliser la commande de suppression. Voici un exemple :

```
file *.txt ¦ less
```

Méfiez-vous de ne pas ajouter d'espace lorsque vous réitérez le masque entre la commande `file` et la commande `rm !`

Méfiez-vous enfin de l'utilisation des métacaractères avec les fichiers cachés. Rappelons qu'un fichier caché commence par un point. Vous pourriez donc croire qu'en écrivant `.*`, vous désignez tous les fichiers cachés. C'est vrai, mais cela désigne aussi la totalité du répertoire courant

(`.`) et son répertoire parent (`..`). Autrement dit, l'écriture `.*` concerne la totalité du contenu du répertoire courant et de son répertoire parent. Bonjour les dégâts !

Supprimer un répertoire (rmdir, rm)

Deux commandes sont disponibles pour supprimer un ou plusieurs répertoires. La première se nomme `rmdir`, et c'est la plus sûre des deux parce qu'elle refuse de fonctionner pour un répertoire qui n'est pas vide (qui contient des fichiers ou des sous-répertoires). Vous l'utilisez en indiquant le nom du répertoire désiré, comme ceci :

```
rmdir bouquins
```

Lorsqu'il vous faut supprimer toute une branche de l'arborescence, la commande `rm` s'avère utile, même si elle est dangereuse. Elle permet de supprimer un répertoire et tout ce qu'il contient, y compris le contenu de ses sous-répertoires, si vous utilisez l'option de récursivité `-R`. Si vous ajoutez l'option de forçage `-f`, tous les fichiers trouvés en chemin sont supprimés aussi. C'est donc une commande dévastatrice mais très efficace. Voici un exemple :

```
rm –Rf bouquins
```

Cerise d'arsenic sur le gâteau, la commande fonctionne en silence. Ici, elle supprime le répertoire bouquins et tout ce qu'il contient.

Vous prendrez soin d'ajouter l'option interactive pour limiter les risques afin que la commande vous demande confirmation pour chaque suppression. Dans l'exemple suivant, nous avions laissé un fichier dans le dossier `montaf/rediger/ bouquins` :

```
pi@raspberrypi ~ $ rm –Rfi montaf
rm: descend into directory 'montaf'? Y
rm: descend into directory 'montaf/rediger'? Y
rm: descend into directory
'montaf/rediger/bouquins'? Y
rm: remove regular file
'montaf/rediger/bouquins/rapidplan.txt'? Y
```

```
rm: remove directory 'montaf/rediger/bouquins'? Y
rm: remove directory 'montaf/rediger'? Y
rm: remove directory 'montaf'? Y
```

Vous pouvez bien sûr utiliser des métacaractères pour supprimer des répertoires, mais vous devez être alors très vigilant et ne pas ajouter d'espaces indésirables qui vous feraient supprimer * (tout). Si vous écrivez par exemple `rm – Rf .*` pour supprimer des répertoires cachés, vous désignez aussi le répertoire courant (.) et son répertoire parent (..). Autrement dit, vous supprimez tout ce que contient le répertoire, caché ou pas, tous les sous-répertoires (avec les fichiers cachés ou pas) et tout ce que contient le répertoire parent avec ses autres sous-répertoires. Horrible efficacité.

Mon expérience de la communauté des utilisateurs Linux m'a prouvé que les gens étaient très accueillants et prêts à vous aider lorsque vous débutez. Pourtant, vous allez parfois rencontrer de petits rigolos qui donnent des conseils erronés aux débutants, en les invitant par exemple à utiliser la commande `rm - Rf /*` en tant que superutilisateur. Résultat net : cela supprime tous les fichiers du système à partir de la racine.

Copier et renommer des fichiers (cp)

Une des opérations que vous aurez besoin de faire le plus fréquemment sur vos fichiers est de les copier. Voyons donc comment faire. La commande correspondante se nomme `cp` et son format générique est le suivant :

```
cp [options] copier_de copier_vers
```

Remplacer *copier_de* par le nom du fichier à copier et *copier_vers* par le nom de la future copie.

Voici par exemple comment copier un fichier nommé *config.txt* se trouvant dans le sous-répertoire */boot* vers votre répertoire personnel (~) pour pouvoir vous amuser avec cette copie sans risque :

```
cp /boot/config.txt ~
```

Pour copier le fichier dans votre répertoire courant, quel que soit son nom réel :

```
cp /boot/config.txt .
```

Vous pouvez spécifier un chemin d'accès vers un dossier existant pour y stocker le fichier :

```
cp /boot/config.txt ~/files/
```

Le fichier ne porte pas nécessairement le même nom dans la copie. Voici par exemple comment recopier un fichier en lui donnant un autre nom :

```
cp /boot/config.txt ~/vieuxconfig.txt
```

Cette commande copie *config.txt* depuis le sous-répertoire */boot* vers votre répertoire personnel sous le nom *vieuxconfig.txt*. Cette technique permet de créer une copie de sécurité d'un fichier sur lequel vous allez travailler, ce qui vous permet de revenir à l'ancienne version au cas où. Les chemins d'accès sont toujours facultatifs. Si vous êtes dans votre répertoire personnel, vous pouvez créer une copie de sécurité au même endroit qu'un fichier nommé *timeplan.txt* ainsi :

```
cp timeplan.txt timeplan.bak
```

La commande `cp` comporte plusieurs options, et la plupart sont les mêmes que pour la commande `rm`. Notez que `cp` écrase le contenu des fichiers sans vous prévenir. Il est donc conseillé d'utiliser l'option `-i` pour demander confirmation avant chaque écrasement par la nouvelle copie. L'option `-v` permet de suivre ce que fait la commande, comme pour `rm`.

Vous pouvez bien sûr utiliser les métacaractères pour copier toute une série de fichiers d'un coup. Pour que la copie se propage aux sous-répertoires, vous ajoutez l'option de récursivité `-R`, comme ceci :

```
cp -R ~/Scratch/* ~/homebak
```

La commande précédente copie tout ce qu'elle trouve dans le répertoire nommé *Scratch* (et tous ses sous-répertoires) vers un répertoire nommé *homebak* de votre répertoire personnel qui doit avoir été créé au préalable. En ce qui concerne la copie vers des périphériques externes, voyez l'Annexe A.

Lorsque vous ne voulez pas copier, mais déplacer un fichier, vous utilisez la commande `mv`. Supposons que vous ayez stocké des images au mauvais

endroit et que vous désirerez les transférer du répertoire *australie* vers le répertoire *japon* qui sont tous deux sous votre répertoire personnel :

```
mv ~/australie/itineraire.txt ~/japon
```

Il faut que le répertoire destinataire existe. Si ce n'est pas le cas, la commande va penser que vous voulez faire une copie du fichier sous le nouveau nom de fichier `japon`. Autrement dit, le fichier qui s'appelait `itinéraire.txt` dans le répertoire `australie` change de nom et devient le fichier nommé `japon` dans le répertoire personnel. Si vous procédez ainsi par erreur, vous êtes perdu, mais c'est ainsi qu'il faut faire pour renommer un fichier sous Linux. En fait, vous déplacez le fichier de l'ancien nom vers le nouveau nom, en général, dans le même dossier. Voici comment :

```
mv vieuxnm nouveaunom
```

Il n'y a pas d'options de récursivité avec la commande `mv` parce qu'elle permet aussi bien de déplacer et de renommer un répertoire qu'un fichier.

Installer et gérer les logiciels du Raspberry Pi

Une fois que vous aurez appris comment procéder, vous constaterez qu'il est très facile sur le Raspberry Pi de découvrir, de télécharger et d'installer de nouveaux logiciels. Les distributions Linux comportent des milliers de paquetages, c'est-à-dire des logiciels préparés pour être téléchargés et installés directement sur votre machine.

Certains paquetages ont besoin d'autres paquetages installés pour bien fonctionner, mais un programme gestionnaire de paquetages gère à votre place tous les problèmes de dépendances. Il se charge de télécharger et d'installer les logiciels dont la présence est obligatoire pour que celui que vous voulez installer fonctionne. Sur le Raspberry Pi, l'outil de gestion des paquetages porte le nom `apt`.

L'installation d'un logiciel suppose de disposer de tous les droits sur la machine, c'est-à-dire d'avoir ouvert la session avec le compte de superutilisateur ou utilisateur *root*. Par mesure de sécurité, le compte de superutilisateur n'est pas actif sur le Raspberry Pi, alors que c'est le cas dans la plupart des autres distributions Linux. En effet, ce compte *root*

peut constituer une brèche de sécurité, parce que les gens ont tendance à utiliser ce compte en permanence au lieu de ne s'y connecter qu'au besoin. De ce fait, le système et tous ses fichiers sont vulnérables aux attaques de logiciels malveillants. Sous Raspberry Pi, vous utilisez de façon transitoire le compte superutilisateur en indiquant le préfixe `sudo` avant chaque commande qui réclame les droits correspondants. Le préfixe n'est pas utilisable devant toutes les commandes, mais il est indispensable pour pouvoir installer un logiciel.

Si vous voyez apparaître un message d'erreur comme quoi ce que vous tentez de faire réclame les droits du superutilisateur, il suffit de répéter la commande en ajoutant le préfixe `sudo` au début.

Actualiser le cache

Avant d'installer tout logiciel, il est bon de commencer par demander une actualisation du référentiel qui est la liste de tous les paquetages logiciels que connaît le gestionnaire. La commande est la suivante :

```
sudo apt-get update
```

Cette commande n'est utilisable que si vous avez une connexion Internet, et vous devez vous montrer patient. Laissez le Raspberry Pi travailler et allez prendre une tasse de thé ou une part de tarte aux framboises.

Trouver le nom d'un paquetage

Le cache référentiel d'apt contient la liste de tous les paquetages logiciels disponibles. Vous pouvez la scruter pour trouver un logiciel. Voici par exemple comment trouver tous les jeux :

```
sudo apt-cache search game
```

La liste étant assez longue, il est conseillé d'ajouter la commande `less` pour paginer l'affichage :

```
sudo apt-cache search game ¦ less
```

Voici le genre de résultat qui s'affiche :

```
pi@raspberrypi ~ $ sudo apt-cache search game
0ad-data - Real-time strategy game of ancient
```

```
warfare
(data)
3dchess - Play chess across 3 boards!
4digits - guess-the-number game, aka Bulls and
Cows
7kaa-data - Seven Kingdoms Ancient Adversaries -
game data
a7xpg - chase action game
a7xpg-data - chase action game - game data
abe - Side-scrolling game named "Abe's Amazing
Adventure"
abe-data - Side-scrolling game named "Abe's
Amazing
Adventure"
[la liste est bien plus longue]
```

Le premier mot dans chaque ligne avant le tiret est le nom officiel du paquetage, qui est celui qu'il faut saisir pour l'installer. Ce nom n'est pas toujours identique au nom complet du logiciel. Il y a par exemple plusieurs jeux de cartes de type solitaire, mais aucun d'entre eux ne porte le nom de paquetage « solitaire ». Pour trouver le nom de paquetage d'un jeu de solitaire, utilisez la commande suivante :

```
sudo apt-cache search solitaire
```

Le résultat comporte une vingtaine de réponses, dont la première est la suivante :

```
ace-of-penguins - penguin-themed solitaire games
```

Installer un logiciel

Dès que vous avez noté le nom du paquetage à installer, vous pouvez utiliser la commande suivante pour demander le téléchargement du paquetage et son installation, ainsi que celle de tous les paquetages dont il dépend pour bien fonctionner (ses dépendances) :

```
sudo apt-get install ace-of-penguins
```

Le quatrième élément dans cette commande est le nom exact du paquetage que nous avons déterminé en effectuant une recherche dans le cache.

Utilisez la commande **apt-cache** pour chercher un logiciel dans le cache local. Pour installer un logiciel, utilisez **apt-get**. Il arrive souvent que l'on intervertisse les deux commandes. Vérifiez ce point si votre commande semble ne pas fonctionner.

Sachez que certains des logiciels proposés ne fonctionnent pas bien sur un nano-ordinateur tel que le Raspberry Pi. Cela dit, rien ne vous empêche d'essayer ceux qui vous intéressent, car la désinstallation est facile.

Lancer un logiciel

Vous pouvez lancer certains programmes directement depuis l'interpréteur en indiquant leur nom :

```
penguinspuzzle
```

La commande précédente lance le jeu *Penguins Puzzle* (c'est d'ailleurs le seul moyen de lancer ce jeu, car il n'est pas présent dans le menu Jeux du Bureau).

La plupart des applications classiques ne fonctionnent que dans l'environnement graphique avec le serveur X Server. Autrement dit, il faut avoir démarré l'environnement de bureau pour les exécuter. Quand vous les aurez installés, même en mode texte, vous pourrez les trouver dans un sous-menu du menu des programmes.

Mettre à jour un logiciel

Le gestionnaire de paquetages permet non seulement d'installer des logiciels, mais également de procéder à leur mise à jour, en installant notamment les améliorations et les correctifs de sécurité. Une seule commande permet de demander la mise à jour de tous les logiciels présents sur votre Raspberry Pi :

```
sudo apt-get upgrade
```

Pensez d'abord à demander l'actualisation du cache pour qu'apt installe les dernières mises à jour des paquetages déjà installés. Vous pouvez combiner les deux commandes en une seule ligne comme ceci :

```
sudo apt-get update && sudo apt-get upgrade
```

Le double symbole & & (*et* commercial) signifie que la deuxième commande ne doit être lancée que si la première a réussi. Si la mise à jour du cache n'a pas réussi, il est inutile de tenter de mettre à jour les logiciels.

Notez que le processus de mise à jour va occuper votre Raspberry Pi pendant quelques minutes éventuellement.

Pour ne mettre à jour qu'une seule application, il suffit d'indiquer la commande qui a permis de l'installer. Supposons que vous ayez déjà installé le jeu *Ace of Penguins*. Saisissez la commande suivante dans ce cas :

```
sudo apt-get install ace-of-penguins
```

Vous demandez ainsi à apt de chercher les mises à jour du paquetage et de les installer s'il y en a. S'il n'y en a pas, il vous confirme que vous disposez de la plus récente version.

Supprimer un logiciel et faire de la place

Le gestionnaire de paquetages permet bien sûr également de supprimer un logiciel, comme ceci :

```
sudo apt-get remove ace-of-penguins
```

Les fichiers techniques de l'application sont tous supprimés, mais pas nécessairement les fichiers créés par l'utilisateur et ses fichiers de paramètres. Si vous êtes certain de ne plus vouloir du tout de ces informations, vous pouvez nettoyer l'espace occupé par une application par la commande de purge suivante :

```
sudo apt-get purge ace-of-penguins
```

Deux autres techniques permettent de libérer de l'espace sur la carte mémoire SD (qui n'est jamais trop spacieuse). Vous pouvez d'abord supprimer automatiquement les paquetages qui ne sont plus nécessaires. En effet, lorsqu'un paquetage est installé, les paquetages dont il dépend sont installés en même temps, et ces derniers peuvent rester en place après

désinstallation du paquetage d'origine. Une commande permet d'éliminer les paquetages dont plus aucun programme n'a besoin. La voici :

```
sudo apt-get autoremove
```

La commande affiche les noms des paquetages qui vont être supprimés et vous indique l'espace qui va être récupéré puis vous demande de confirmer par **Y** que vous voulez continuer.

Lorsque vous demandez l'installation d'un paquetage, le fichier archive du paquetage est d'abord récupéré puis stocké sur votre Raspberry Pi dans le répertoire `/var/cache/apt/archives`. Une fois que vous aurez installé un certain nombre de paquetages, cela peut représenter un certain espace sur la carte SD. Allez donc jeter un œil sur le contenu de ce répertoire. Ces fichiers n'ont plus aucun intérêt. Si vous réinstallez un programme, vous pouvez toujours télécharger le même fichier.

La deuxième technique permettant de libérer de l'espace sur la carte SD consiste à supprimer tous ces fichiers d'archives ainsi :

```
sudo apt-get clean
```

Lister les programmes installés (dpkg)

Vous pouvez savoir à tout moment quels sont les logiciels installés sur votre Raspberry Pi avec la commande suivante :

```
dpkg --list
```

Cette commande ne pouvant pas avoir de fâcheuses conséquences, vous n'êtes pas forcé d'ajouter la mention `sudo` en préfixe.

Pour savoir si un paquetage en particulier est bien installé, utilisez cette commande :

```
dpkg --status nompaquetage
```

Cette commande fournit une description plus élaborée que celle offerte par `apt cache`, et notamment un lien Web pour obtenir d'autres informations.

Le Raspberry Pi est livré avec de nombreux paquetages consécutifs du système d'exploitation Linux. Soyez donc très vigilant avant de décider

de supprimer un paquetage que vous n'avez pas installé vous-même.

Gérer les comptes utilisateurs

Aussi petit soit-il, vous pouvez partager votre Raspberry Pi avec d'autres utilisateurs ou membres de votre famille en créant un compte pour chacun. De ce fait, chacun aura son propre répertoire personnel. Le mécanisme de droits d'accès très rigoureux de Linux garantit qu'aucun utilisateur ne pourra effacer par mégarde les fichiers d'un autre.

Nous avons présenté les droits d'accès un peu plus haut dans ce chapitre. Vous vous souvenez qu'un utilisateur était membre d'au moins un groupe. Sur le Raspberry Pi, les groupes servent à contrôler l'accès aux ressources comme le matériel audio ou vidéo. Avant de créer un nouveau compte, il faut donc savoir à quel groupe il est utile d'associer le nouvel utilisateur. Pour en savoir plus, utilisez d'abord la commande `groups` qui affiche les noms des groupes dont fait partie par défaut l'utilisateur `pi` :

```
pi@raspberrypi ~ $ groups pi
pi : pi adm dialout cdrom sudo audio video
plugdev games users netdev input
```

Lorsque vous allez créer un compte utilisateur, vous l'associerez avec la plupart des groupes mentionnés ci-dessus, sauf bien sûr le groupe `pi` (qui est celui du premier utilisateur `pi`). Par ailleurs, sachez que si vous associez un utilisateur au groupe spécial `sudo`, il pourra installer des logiciels, modifier tous les mots de passe et faire tout ce qu'il veut sur la machine (s'il sait comment faire). Sur une machine domestique, cela ne devrait pas poser de problèmes. Le système des droits d'accès protège malgré tout les utilisateurs des effacements accidentels, tant qu'ils restent vigilants quand ils utilisent le préfixe `sudo`.

Pour créer un compte utilisateur, vous vous servez de la commande `useradd` avec l'extension `-m` qui demande la création d'un répertoire personnel et l'option `-G` pour afficher la liste des groupes dont l'utilisateur doit devenir membre :

```
sudo useradd –m –G [liste de groupes] [nomutil]
```

Voici un exemple réel :

```
sudo useradd –m –G
```

```
adm,dialout,cdrom,sudo,audio,video,plug
dev,games,users,netdev,input karen
```

Notez bien que les noms des groupes sont tous séparés par une virgule, mais sans aucune espace. La première espace précède le nom choisi pour le nouvel utilisateur.

Une fois cette commande émise, vous pouvez vérifier qu'un nouveau répertoire personnel a été créé avec le nom de l'utilisateur. Rendez-vous dans le répertoire `/home` ou demandez l'affichage de son contenu actuel :

```
ls /home
```

Vous devez ensuite définir un mot de passe pour le nouveau compte :

```
sudo passwd [nomutil]
```

Voici un exemple concret :

```
sudo passwd karine
```

Les majuscules et les minuscules sont distinguées dans les identifiants. La commande vous demande de saisir deux fois le mot de passe pour être certain que vous n'avez pas fait de faute de frappe. Vous ne pouvez pas voir ce que vous frappez pour le mot de passe. En cas de souci, vérifiez d'abord que le clavier est bien en français et que la touche de verrouillage Maj n'est pas active.

Pour vérifier que tout fonctionne, il suffit d'essayer de se reconnecter avec ce nouveau compte. Fermez la fenêtre du Terminal puis ouvrez le menu général et choisissez **Shutdown** puis **Logout**. Vous fermez ainsi la session avec votre compte d'utilisateur normal (**pi**). Le système vous demande de montrer patte blanche avec le nouvel identifiant et son mot de passe. Faites ensuite de même pour revenir à votre compte normal **pi**. Rappelons que le mot de passe du compte normal est **raspberry**.

Si vous êtes en mode texte (sans Bureau démarré), il suffit de se déconnecter du compte courant avec la commande suivante :

```
logout
```

Si vous vous servez de la commande `passwd` pour définir un mot de passe pour l'utilisateur spécial nommé *root*, vous pouvez ensuite vous connecter à la machine en tant que superutilisateur et disposer de tous les

pouvoirs sur la machine. Cela peut vous permettre de faire fonctionner certains logiciels, mais nous vous déconseillons d'utiliser ce compte. Mieux vaut n'accédez aux droits de superutilisateur que de façon temporaire au moyen du préfixe `sudo`.

Rappelons que de nos jours une carte mémoire SD de taille suffisante coûte une poignée d'euros. Vous pouvez donc partager le même Raspberry Pi entre plusieurs membres de la famille en préparant une carte SD système pour chacun. Chacun se connectera en utilisant le compte initial *pi* avec le mot de passe associé.

Se documenter sur les commandes

Internet regorge d'informations au sujet de Linux, mais le système en héberge aussi une certaine quantité. Pour aller plus loin dans la maîtrise de Linux, cette documentation, bien qu'en anglais, peut vous donner quelques indices, mais le langage est parfois un peu technique.

Sous Linux, une commande peut correspondre à différents formats. Soit il s'agit d'un sous-programme intégré à l'interpréteur (une primitive), soit d'un fichier de programme indépendant que vous trouverez dans ce cas dans le sous-répertoire */bin*, soit il s'agit d'un alias (nous nous en expliquons dans la section suivante). Pour accéder à la documentation texte d'une commande, il faut d'abord savoir quel est le type de la commande au moyen de la commande `type` :

```
pi@raspberrypi ~ $ type cd
cd est une primitive du shell
pi@raspberrypi ~ $ type mkdir
mkdir est /bin/mkdir
pi@raspberrypi ~ $ type ls
ls est un alias vers 'ls --color=auto'
```

Pour savoir où se trouve le fichier d'un programme de commande, vous vous servez de la commande `which` en indiquant le nom de la commande :

```
which mkdir
```

Pour accéder à la documentation des commandes intégrées à l'interpréteur, vous utilisez l'outil `help` de celui-ci. Indiquez le nom `help` suivi du nom de fichier désiré.

```
help cd
```

Dans la documentation de `help`, des crochets droits désignent les différentes options facultatives et la barre verticale sépare les éléments qui s'excluent mutuellement, tels que des options contraires.

En ce qui concerne les programmes indépendants comme `mkdir`, vous pouvez lancer la commande avec l'option `– help`. La plupart des programmes sont prévus pour reconnaître cette option et affichent une documentation minimale. Voici un exemple :

```
mkdir --help
```

Cette documentation n'est pas dénuée d'humour. Essayez par exemple d'accéder à l'aide d'`apt-get`. Essayez par exemple `apt-get moo` pour voir !

La plupart des programmes sont dotés d'un véritable manuel que l'on appelle une page *man*, et c'est aussi le cas des commandes Linux de type programme et de quelques applications comme LibreOffice (voir Chapitre 6). Pour accéder au manuel d'un programme, saisissez :

```
man nom_programme
```

Voici un exemple :

```
man ls
```

L'affichage utilise d'office la commande de pagination `less`, ce qui vous permet d'utiliser les touches auxquelles cette commande vous a habitué. Notez que la documentation peut avoir une tournure assez technique, elle n'est donc pas aussi digeste que celle des pages *help*.

Lorsque vous cherchez une commande, vous pouvez faire chercher un terme anglais parmi toutes les pages des manuels au moyen de la commande `apropos` (sans accent) :

```
pi@raspberrypi ~ $ apropos delete
argz_delete (3) - functions to handle an argz
list
delete_module (2) - delete a loadable module
entry
dphys-swapfile (8) - set up, mount/unmount, and
```

```
delete an swap file
groupdel (8) - delete a group
rmdir (2) - delete a directory
shred (1) - overwrite a file to hide its
contents, and optionally delete it
tdelete (3) - manage a binary tree
timer_delete (2) - delete a POSIX per-process
timer
tr (1) - translate or delete characters
unlink (2) - delete a name and possibly the file
it refers to
userdel (8) - delete a user account and related
files
```

Parmi les programmes que cite la commande, vous pouvez chercher de façon plus détaillée dans les pages *man* ou voir si une commande accepte l'option `--help`. Le chiffre entre parenthèses correspond à la section du manuel contenant le mot que vous avez recherché.

Pour n'afficher qu'un résumé sur une ligne du manuel, utilisez la commande `whatis` :

```
pi@raspberrypi ~ $ whatis ls
ls (1) - list directory contents
```

Vous en voulez encore plus ? Vous disposez également en plus de la page *man* de la page *info*. Ce genre de page se présente un peu comme un site Web, avec un sommaire au départ et des liens vers les différentes pages. Voici comment utiliser cette commande :

```
info ls
```

Les touches pour se déplacer dans un document *info* ne sont pas toutes les mêmes que pour une page *man*. Pour connaître la liste des touches utilisables, appuyez sur le point d'interrogation lorsqu'une page info est ouverte.

Définir une nouvelle commande Linux

Pour personnaliser encore plus votre Raspberry Pi, sachez que vous pouvez définir de nouvelles commandes Linux. Vous pouvez par exemple créer une commande qui affiche un message spécial dès que quelqu'un saisit votre nom (en vous servant de la commande `echo`). Plus sérieusement, vous pouvez définir des raccourcis pour englober une commande avec un jeu d'options que vous utilisez souvent. Nous allons vous montrer comment définir une commande permettant de supprimer des fichiers en appliquant d'office les options conseillées pour confirmer avant chaque suppression et indiquer ce qui a été réalisé. Nous allons baptiser cette nouvelle commande `pidel`, qui rappelle pi et delete.

Tout d'abord, il y a lieu de vérifier si ce nom choisi pour la nouvelle commande n'est pas déjà utilisé. Si la commande `type` indique qu'il n'existe pas de fichier portant ce nom, tout va bien. Dans le cas contraire, changez de nom pour ne pas vous voir interdire l'accès à une commande existante. Voici mon test :

```
pi@raspberrypi ~ $ type pidel
-bash: type: pidel: not found
```

Une fois que vous avez vérifié que la commande `pidel` n'existait pas encore, nous pouvons la créer. Nous allons définir un alias comme ceci :

```
alias pidel='rm –vi'
```

La commande Linux et toutes les options sont placées entre apostrophes en équivalence du nom de notre nouvelle commande. Dorénavant, lorsque vous écrivez `pidel`, cela revient à écrire `rm -vi`, mais vous n'allez plus à vous soucier d'indiquer les options. Voici un exemple d'utilisation :

```
pi@raspberrypi ~ $ pidel *.txt
rm: remove regular file 'fm.txt'? y
removed 'fm.txt'
rm: remove regular file 'toc.txt'? n
pi@raspberrypi ~ $
```

Dans une définition d'alias, vous pouvez indiquer plusieurs commandes en les séparant par des points-virgules, comme ceci :

```
alias pidel='clear;echo Supprime des fichiers
avec les options interactive et
```

```
verbeuse.;rm –vi'
```

Notez que les alias sont perdus lorsque vous éteignez la machine. Pour les rendre permanents, il faut stocker l'instruction `alias` dans le fichier caché *.bashrc* de votre répertoire personnel. Vous devez d'abord charger ce fichier dans un éditeur, comme ceci :

```
nano ~/.bashrc
```

Nano est un éditeur de texte très simple dont nous parlons dans l'Annexe A. Sachez qu'il suffit d'ajouter l'alias dans ce fichier, d'enregistrer avec le raccourci Ctrl + O puis de quitter avec Ctrl + X.

Vous pouvez placer l'alias n'importe où dans le fichier *.bashrc*. Pour éviter de perturber le contenu actuel du fichier, nous vous conseillons de l'insérer tout en haut. Chaque alias doit être sur une ligne indépendante.

Il ne reste plus qu'à vérifier que votre nouvelle commande est bien disponible en permanence. Éteignez et rallumez votre Raspberry Pi ou provoquez un démarrage à chaud (voir plus bas).

Dans certains cas, vous voudrez masquer une commande existante en définissant un alias homonyme. De cette manière, les options que vous précisez seront utilisées en priorité. Si vous regardez le type de la commande `ls`, vous constaterez que c'est un alias qui permet d'appliquer des options de couleurs à l'affichage des fichiers.

Arrêter et redémarrer le système

Le Raspberry Pi consommant extrêmement peu, vous pouvez le laisser en veille entre deux séances. En revanche, avant de brancher ou débrancher quoi que ce soit et avant un arrêt prolongé, prenez la bonne habitude d'arrêter votre système.

Pour arrêter et redémarrer votre Raspberry Pi, vous utilisez les commandes du sous-menu **Shutdown** du Bureau.

La répétition des opérations de branchement et débranchement du connecteur d'alimentation finit par user les points de soudure du Raspberry. Il est vraiment conseillé de s'équiper d'une multiprise avec interrupteur. Lorsque vous éteignez pour rallumer, attendez quelques secondes dans l'état éteint que les condensateurs aient le temps de se décharger.

Si vous êtes en mode texte (sans interface graphique), émettez la commande suivante afin que le système procède à la purge des fichiers temporaires et à l'arrêt des services :

```
sudo halt
```

Toujours en mode texte, vous demandez un redémarrage à chaud avec cette commande :

```
sudo reboot
```

PARTIE 3
Travailler et se cultiver avec le Raspberry Pi

DANS CETTE PARTIE :

- Avec LibreOffice, rédigez vos courriers, gérez votre budget, créez des diaporamas et dessinez une invitation pour une fête.
- Avec GIMP, retouchez vos photos, en faisant des rotations et des redimensionnements, en supprimant les petits défauts et en rognant vos images.
- Visionnez des films en haute définition et écoutez de la musique sur votre Raspberry Pi grâce au système LibreELEC qui le transforme en un centre multimédia de salon.

Chapitre 6

La bureautique avec le Raspberry Pi

DANS CE CHAPITRE :

- **Installer LibreOffice sur votre Raspberry Pi**
- **Rédiger un courrier avec LibreOffice Writer**
- **Gérer votre budget avec LibreOffice Calc**
- **Créer un diaporama avec LibreOffice Impress**
- **Créer une invitation avec LibreOffice Draw**

Tout le monde a besoin à un moment ou à un autre de passer aux choses sérieuses. Le Raspberry Pi peut vous aider à réaliser les activités de base d'un humain moderne que sont l'écriture, le calcul et le dessin. Que vous ayez besoin de vaquer à vos tâches d'administration domestiques ou professionnelles, vous pouvez utiliser LibreOffice, une suite bureautique complète qui fonctionne sur le Raspberry Pi.

Si vous n'avez jamais rencontré le nom LibreOffice, il est possible que vous ayez entendu parler de son prédécesseur plus connu : OpenOffice. En fait, un groupe de développeurs passionnés est reparti du code source d'OpenOffice pour créer LibreOffice.

Les domaines d'emploi et le mode d'utilisation de LibreOffice sont très proches de ceux de Microsoft Office pour Windows. LibreOffice ne vous posera sans doute aucun problème de prise en main. Les fichiers sont compatibles entre les deux suites, même s'il faut reconnaître que quelques éléments pointus de mise en page sont parfois perdus en cours de route.

Je vous propose dans ce chapitre de découvrir quatre des programmes essentiels de LibreOffice qui vous seront utiles pour vos activités habituelles. Vous allez découvrir comment rédiger un courrier, comment

utiliser le tableur pour faire un budget, comment créer un diaporama ou une présentation et comment dessiner un carton d'invitation.

LibreOffice est totalement gratuit ; vous pouvez le télécharger et même le redistribuer. Si après avoir essayé le programme, vous pensez que cet énorme effort mérite d'être soutenu, vous pouvez faire un don à l'association The Document Foundation qui pilote son développement en vous rendant sur son site Web : `www.libreoffice.org`.

Installation de LibreOffice sur le Raspberry Pi

LibreOffice est normalement préinstallé dans Raspbian. Si ce n'est pas le cas, vous pouvez télécharger et installer LibreOffice très facilement en émettant successivement les deux commandes suivantes dans une fenêtre de terminal ou sur la ligne de commande de Linux :

```
sudo apt-get update
sudo apt-get install libreoffice
```

Pour tous autres détails concernant l'installation des logiciels et sur la signification de ces deux commandes, nous vous invitons à revenir au Chapitre 5.

Démarrage de LibreOffice

Lorsque vous êtes face au bureau, vous pouvez ouvrir le menu Programmes pour trouver dans la catégorie Bureautique les différents programmes de la suite LibreOffice (Figure 6.1). Vous y trouvez LibreOffice Base (pour les bases de données), LibreOffice Calc (le tableur), LibreOffice Draw (le dessin et la mise en page), LibreOffice Impress (les diaporamas) et LibreOffice Writer (le traitement de texte).

En ouvrant le menu Fichier, vous pouvez démarrer la création d'un fichier LibreOffice de n'importe quel format. Vous pouvez même, lorsque vous êtes dans le traitement de texte, ouvrir le menu Fichier et choisir de démarrer la création d'une feuille de calcul. L'application appropriée est démarrée sur un fichier vide.

Figure 6.1 : Le sous-menu d'accès aux modules de LibreOffice.

Nous ne présentons dans ce chapitre que les quatre applications principales Writer, Calc, Impress et Draw.

Rappelons que vous pouvez également démarrer LibreOffice en chargeant directement un fichier : il suffit de double-cliquer dans le nom du fichier LibreOffice ou Microsoft Office dans une fenêtre du Gestionnaire de fichiers.

Si votre projet de rédaction requiert l'écriture de formules mathématiques, sachez que la suite comporte un module nommé LibreOffice Math qui permet de bien les mettre en forme (mais le programme ne résout pas les équations à votre place). Pour y accéder, il suffit de choisir Formule dans la fenêtre de LibreOffice.

Sauvegarder le travail

Dans tous les modules de LibreOffice, vous enregistrez sur disque le travail actuel en ouvrant le menu **Fichier**, dans lequel vous trouvez la commande **Enregistrer**. De nombreux formats sont disponibles, mais le format par défaut est le nouveau format universel Open Document Format, ODF (qui est devenu lisible par la dernière version de Microsoft Office). Vous pouvez aussi enregistrer dans les formats anciens de Microsoft Office (.doc, .xls, *etc.*).

LibreOffice effectue automatiquement une sauvegarde périodique et la suite est dotée d'un mécanisme très efficace pour récupérer des fichiers corrompus suite à un blocage de l'ordinateur. Cela dit, mieux vaut rester prévoyant. Prenez donc la saine habitude d'enregistrer fréquemment votre travail.

Rédiger un courrier avec Writer

LibreOffice Writer est un traitement de texte très proche de Microsoft Word pour Windows. C'est donc l'application à utiliser pour rédiger des courriers et des textes.

Le programme sait ouvrir des fichiers au format Microsoft Word. Depuis quelques années, le format standard ODF (extension .odt) peut être lu et écrit par Word. Ne soyez pas surpris de constater, lorsque vous ouvrez un document Microsoft Word dans LibreOffice, de voir quelques perturbations au niveau des détails de la mise en page. La première cause en sera qu'en général vous ne disposerez pas des mêmes polices de caractères que sur un PC dans votre Raspberry Pi. De plus, les options de mise en page Word sophistiquées sont interprétées différemment.

La Figure 6.2 donne un aperçu de LibreOffice Writer en pleine action. Si vous avez déjà utilisé un traitement de texte, il ne vous faudra pas longtemps pour être à l'aise. Les icônes ressemblent beaucoup à celles qu'utilise Microsoft Office. Lorsque vous amenez le pointeur sur une icône sans cliquer, vous voyez apparaître une bulle d'aide qui vous donne une explication.

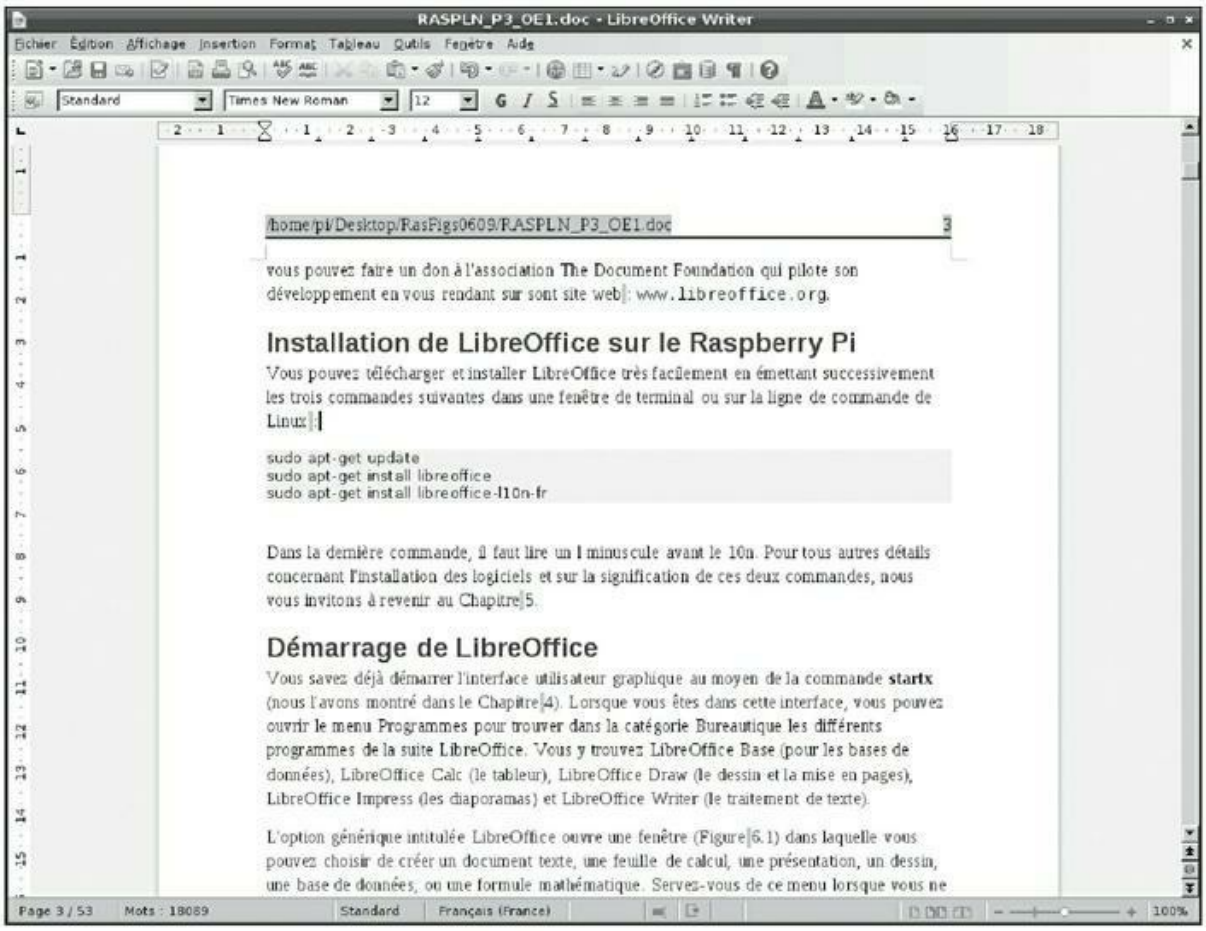

RASPLN_P3_OE1.doc - LibreOffice Writer
Fichier Édition Affichage Insertion Format Tableau Outils Fenêtre Aide
Standard
Times New Roman
12
/home/pi/Desktop/RasPigs0609/RASPLN_P3_OE1.doc 3
vous pouvez faire un don à l'association The Document Foundation qui pilote son développement en vous rendant sur sont site web : www.libreoffice.org.
Installation de LibreOffice sur le Raspberry Pi
Vous pouvez télécharger et installer LibreOffice très facilement en émettant successivement les trois commandes suivantes dans une fenêtre de terminal ou sur la ligne de commande de Linux :
sudo apt-get update
sudo apt-get install libreoffice
sudo apt-get install libreoffice-l10n-fr
Dans la dernière commande, il faut lire un l minuscule avant le 10n. Pour tous autres détails concernant l'installation des logiciels et sur la signification de ces deux commandes, nous vous invitons à revenir au Chapitre 5.
Démarrage de LibreOffice
Vous savez déjà démarrer l'interface utilisateur graphique au moyen de la commande startx (nous l'avons montré dans le Chapitre 4). Lorsque vous êtes dans cette interface, vous pouvez ouvrir le menu Programmes pour trouver dans la catégorie Bureautique les différents programmes de la suite LibreOffice. Vous y trouvez LibreOffice Base (pour les bases de données), LibreOffice Calc (le tableur), LibreOffice Draw (le dessin et la mise en pages), LibreOffice Impress (les diaporamas) et LibreOffice Writer (le traitement de texte).
L'option générique intitulée LibreOffice ouvre une fenêtre (Figure 6.1) dans laquelle vous pouvez choisir de créer un document texte, une feuille de calcul, une présentation, un dessin, une base de données, ou une formule mathématique. Servez-vous de ce menu lorsque vous ne
Page 3 / 53
Mots : 18089
Standard
Français (France)
100%

Figure 6.2 : Rédaction d'une lettre avec LibreOffice.

Pour changer l'aspect et le style du texte, vous vous servez des icônes et des options de la barre de menus au-dessus du document. Le texte que vous saisissez ensuite adopte les attributs choisis. Vous pouvez également sélectionner une partie du texte par glisser-déplacer avant de choisir les options de format que vous voulez appliquer à cette sélection.

Les menus déroulants en haut de fenêtre donnent accès à toutes les possibilités de Writer. Il est conseillé de les passer en revue pour se faire une idée de tout ce que l'application autorise.

Par exemple, le menu **Insertion** vous permet d'insérer des caractères spéciaux qu'il n'est pas possible de saisir au clavier, des sauts manuels (de page ou de ligne), des marques de formatage (par exemple des tirets insécables), l'en-tête ou le pied de document, des notes de bas de page, des signets (ils n'apparaîtront pas dans le document, mais facilitent la navigation), des commentaires (pour un texte rédigé à plusieurs), des cadres (des boîtes dans lesquelles vous pouvez saisir du texte ou insérer des images et que vous pouvez repositionner) et même des tableaux.

Le menu **Format** réunit toutes les options pour contrôler le format des caractères (nom de la police, attributs, soulignement, mise en exposant, rotation, création d'un lien et couleur d'arrière-plan), le format du

paragraphe (indentation et retrait, alignement, habillage du texte et bordure), l'ajout de puces et de numéros automatiques, le format général de la page (format de papier, couleur ou image d'arrière-plan, en-tête et pied), la création de plusieurs colonnes et l'alignement.

L'essentiel de vos besoins correspond à ces deux menus. Les commandes les plus fréquemment utilisées font l'objet d'une icône sous la barre de menus en haut de fenêtre.

Si vous prenez la bonne habitude d'utiliser des styles de paragraphes pour structurer votre document (par exemple en appliquant le style Titre 1 pour les grands titres et Titre 2 pour les sous-titres), cela vous permet ensuite d'ouvrir la fenêtre du Navigateur (raccourci F5) pour voir directement la structure de votre document. Vous pouvez même cliquer pour vous rendre à la section correspondante et passer ainsi directement vers les tableaux, les liens et les signets.

Le menu **Fichier** comporte une option d'exportation au format PDF. Rappelons que le format .pdf permet de protéger l'aspect visuel d'un document pour le diffuser sans crainte auprès des personnes qui ne disposent pas des polices ni du logiciel de création. Vous êtes assuré que tout le monde verra votre texte comme vous le voyez. Tout le monde dispose sur son ordinateur d'un logiciel lecteur de fichiers PDF, mais beaucoup plus rares sont ceux qui peuvent modifier un tel fichier. Ce format est donc d'abord destiné à la diffusion de documents qui ne doivent plus être modifiés par la suite, mais seulement lus.

Gérer un budget avec Calc

LibreOffice Calc, le tableur de la suite, ressemble beaucoup à Microsoft Excel. D'ailleurs, une bonne manière de le découvrir consiste à lui faire charger un fichier de feuille de calcul Excel. Toutes vos formules doivent fonctionner correctement et les formats de mise en page des cellules devraient avoir été préservés (seules les macros ne sont pas compatibles). L'interface ressemble beaucoup à celle de LibreOffice Writer. Vous disposez d'une série d'icônes dotées d'une bulle d'aide. La Figure 6.3 donne en exemple une feuille de calcul de budget de vacances. Nous avons utilisé le curseur de réglage du taux de zoom dans le bas à droite de la fenêtre pour rendre le texte plus lisible.

Les feuilles de calcul Microsoft Excel contenant des macros sophistiquées ne seront pas compatibles avec LibreOffice Calc.

Ce livre ne permet pas d'entrer dans le détail de l'utilisation d'un tableur, mais nous avons néanmoins assez de place pour découvrir comment créer

un budget simplifié avec Calc.

Une feuille de calcul est un tableau de lignes et de colonnes constituées de cellules. Sa puissance est liée au fait que vous pouvez non seulement insérer des données dans les cellules, mais également des formules qui font référence à d'autres cellules. Pour saisir quelque chose dans une cellule, il suffit de cliquer dans la cellule puis de commencer la saisie. Vous pouvez aussi cliquer dans la cellule puis effectuer la saisie dans la barre de formule en haut de fenêtre (voir Figure 6.3).

Chaque cellule possède une référence construite à partir de la lettre de la colonne (voir la ligne d'intitulé) et du numéro de la ligne (voir à gauche). La cellule de l'angle supérieur gauche correspond à A1, et sa voisine de droite est la B1, celle en dessous est la B2, et ainsi de suite (Figure 6.3).

Commencez par saisir en vous inspirant des intitulés de la figure des différents postes de dépenses dans la colonne A. Dans la colonne B, saisissez le montant de chaque dépense et dans la colonne C la quantité pour chaque poste. Dans notre exemple, il y a un poste qui correspond à l'hôtel avec son coût par nuit dans la colonne B et la valeur 6 pour le nombre de nuitées dans la colonne C. Vous remarquez dans la Figure 6.3 que nous avons pris soin d'ajouter une ligne d'intitulé pour les différentes colonnes, ce qui permet de communiquer plus aisément les données saisies.

Vous pouvez facilement augmenter la largeur d'une colonne pour voir la totalité du texte que vous y avez saisi. Il suffit d'amener le pointeur sur une barre séparatrice dans la ligne des intitulés de colonnes puis de cliquer et de déplacer vers la droite.

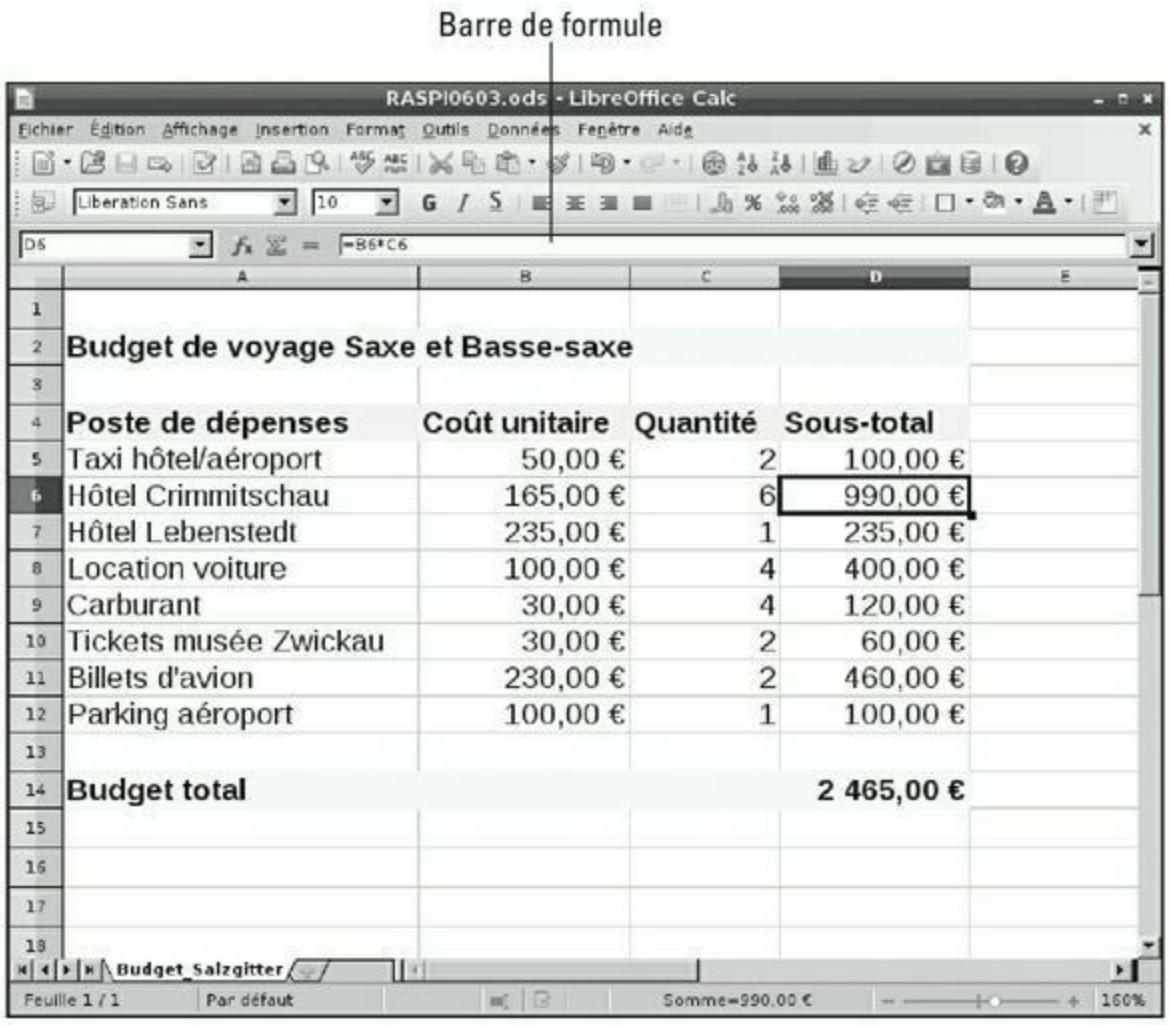

Barre de formule
RASPI0603.ods - LibreOffice Calc
Fichier Édition Affichage Insertion Format Outils Données Fenêtre Aide
Liberation Sans
10
D6
=B6*C6
Budget de voyage Saxe et Basse-saxe
Poste de dépenses
Coût unitaire
Quantité
Sous-total
Taxi hôtel/aéroport 50,00 € 2 100,00 €
Hôtel Crimmitschau 165,00 € 6 990,00 €
Hôtel Lebenstedt 235,00 € 1 235,00 €
Location voiture 100,00 € 4 400,00 €
Carburant 30,00 € 4 120,00 €
Tickets musée Zwickau 30,00 € 2 60,00 €
Billets d'avion 230,00 € 2 460,00 €
Parking aéroport 100,00 € 1 100,00 €
Budget total 2 465,00 €
Budget_Salzgitter
Feuille 1 / 1
Par défaut
Somme=990,00 €
100%

Figure 6.3 : Feuille de calcul de budget de vacances avec LibreOffice Calc.

Vous voulez faire afficher des symboles monétaires ? Ouvrez le menu **Format**, choisissez **Cellule** puis choisissez la catégorie monétaire et le sous-format ainsi que le symbole des devises désirées. Vous pouvez appliquer cette opération en sélectionnant d'abord tout un groupe de cellules. Cliquez et déplacez la sélection pour englober les cellules désirées avant d'ouvrir le menu **Format**.

Nous avons dit que vous pouvez saisir une formule (ou un calcul) dans une cellule, ce qui va faire apparaître le résultat à la place de la formule. Pour y parvenir, il faut commencer la saisie par le signe égal (=). La multiplication correspond au symbole astérisque (*) et la division à la barre oblique (/). En guise d'exemple, cliquez dans une cellule vide et saisissez ceci :

```
=7*5
```

Dès que vous validez, vous voyez apparaître le résultat (35) à la place de ce que vous avez saisi dans la cellule. Pour voir la formule ou la modifier,

cliquez dans cette cellule puis observez la barre de formule au-dessus de la feuille de calcul. Vous pouvez aussi double-cliquer dans la cellule.

Les possibilités sont décuplées à partir du moment où vous citez l'adresse d'une autre cellule dans la formule. Il suffit d'indiquer les coordonnées de la cellule. Dans notre exemple, nous avons besoin de multiplier le coût unitaire de chaque poste (une nuit d'hôtel) par le nombre de nuits prévues (six nuits dans l'exemple).

Le coût unitaire est stocké dans la colonne B et la quantité dans la colonne C de la même ligne. Le premier poste de dépense est situé après le titre et la ligne d'intitulé, en ligne 5. Cliquez dans la cellule D5 pour saisir ceci :

```
=B5*C5
```

Vous demandez la multiplication de la valeur lue dans la cellule B5 (le prix unitaire) par la valeur lue dans la cellule C5 (la quantité), le résultat étant stocké dans la cellule contenant la formule, c'est-à-dire la cellule D5. Mieux encore : la copie est intelligente. Cliquez dans la cellule D5 puis copiez son contenu avec le raccourci Ctrl + C (ou l'option correspondante du menu **Édition**). Placez-vous dans la cellule D6 et collez le contenu du presse-papiers par le raccourci Ctrl + V.

Vous pourriez croire que la valeur affichée dans la première cellule va être copiée dans l'autre, mais c'est la formule qui est copiée. De plus, elle est copiée après adaptation des coordonnées des cellules référencées. Lorsque vous copiez la formule de la cellule D5 dans la cellule D6 puis cliquez dans D6 pour voir la formule, voici ce que vous observez :

```
=B6*C6
```

C'est effectivement ce que nous voulions. Copiez de cette manière la formule dans toutes les autres lignes de cette colonne D afin de disposer du sous-total de chaque poste. Il ne reste plus qu'à ajouter une formule pour le total général. Nous allons utiliser à cet effet une formule spéciale utilisant une fonction nommée SOMME. Son rôle est d'additionner toutes les valeurs d'une plage de cellules spécifiées. Voici comment saisir cette formule :

1. **Cliquez dans une cellule libre sous les sous-totaux en colonne D et commencez la saisie ainsi :**

```
=SOMME(
```

Ne validez pas la saisie en frappant la touche Entrée.

2. **Cliquez dans la première cellule de la colonne de sous-totaux (en D5) et maintenez l'appui sur le bouton de souris**.

3. **Déplacez le pointeur jusqu'à voir le cadre rouge délimiter tous les sous-totaux**.

4. **Relâchez le pointeur de souris puis saisissez une parenthèse fermante) avant de valider par Entrée**.

Le total général apparaît dans la cellule saisie et le budget est terminé. Vous constatez qu'un tableur est bien plus qu'une calculatrice sophistiquée. Vous pouvez vous en servir pour planifier et réaliser des simulations, par exemple pour faire recalculer le total en changeant uniquement le prix de l'hôtel. Toutes les cellules dont le contenu dépend de celle qui a été modifiée sont mises à jour automatiquement, y compris le grand total. Vous pouvez aussi décider de rester plus longtemps en vacances en changeant le nombre de nuits dans la colonne B. Vous voyez immédiatement l'effet sur le budget.

Créer un diaporama avec Impress

Que vous ayez à préparer une présentation pour une réunion ou que vous vouliez faire admirer vos photos de vacances dans un diaporama, vous disposez de LibreOffice Impress pour créer un diaporama puis le lire. Vous savez que tous les modules de LibreOffice ont un équivalent dans la suite bureautique Office de Microsoft. Impress ressemble à Microsoft PowerPoint. Il permet de charger des fichiers PowerPoint, et la perte au niveau des options de format sophistiquées est pratiquement imperceptible, sauf pour certaines transitions entre diapositives. La seule remarque est qu'à chaque image est associé un texte conteneur qui est masqué dans PowerPoint mais qui apparaît dans Impress.

La Figure 6.4 donne un aperçu de l'élaboration d'un diaporama en cours. Voici comment très simplement créer votre propre présentation :

1. **Démarrez le module Impress, ou bien choisissez depuis le menu Fichier de n'importe**

quel autre module LibreOffice de créer une nouvelle présentation.

2. **Dans le volet Tâches, cliquez dans la catégorie Mises en page**.

   Vous voyez apparaître un panneau qui propose 12 mises en page prédéfinies.

3. **Choisissez le modèle dont vous voulez partir**.

4. **Cliquez dans la boîte de titre pour choisir le titre de la diapositive**.

5. **Vous constatez que la diapositive offre six boîtes de contenu**. Cliquez dans une boîte et commencez à saisir.

   Au centre de chaque boîte de contenu se trouvent quatre boutons qui permettent d'insérer un contenu spécifique : un tableau, un diagramme, une image ou une vidéo. Pour rajouter une image, cliquez le bouton en bas à gauche dans ce groupe et désignez le fichier d'image à insérer. Notez que si vous cliquez dans un autre modèle de mise en page à droite, ce modèle va être appliqué à la diapositive en cours d'élaboration.

6. **Pour rajouter une deuxième diapositive, utilisez le bouton Diapos de la barre de menus (voyez dans la Figure 6.4) ou servez-vous du menu Insertion.**

   Répétez les Étapes 3 à 5 pour remplir la diapositive.

7. **Pour modifier une diapositive existante, sélectionnez-la d'abord dans le volet des diapos**

du côté gauche.

Pour enrichir ensuite l'aspect et le format d'un titre, d'un texte ou d'une image, sélectionnez d'abord l'élément dans la zone de la diapo puis utilisez les options de la barre de menus et de la barre d'icônes en haut de fenêtre. Le contenu de ces deux barres varie en fonction de la nature de l'élément sélectionné.

Prenez soin de ne pas insérer des images trop volumineuses, car elles ralentissent l'ordinateur et peuvent même bloquer le logiciel si vous en insérez trop. Le Chapitre 7 donne des détails concernant la manière de redimensionner une photo numérique ou toute autre image.

Figure 6.4 : Création d'un diaporama avec LibreOffice Impress.

Lorsque vous amenez le pointeur dans les limites d'une des diapos déjà créées dans le panneau de gauche, vous voyez apparaître trois boutons, qui permettent de démarrer le diaporama, de masquer la diapo (elle n'est pas supprimée, seulement masquée dans le diaporama) ou de faire une copie de la diapo. Le lancement de diaporama est également possible depuis le menu **Diaporama** en haut de fenêtre ou par frappe de F5. Si vous voulez démarrer depuis la première diapo, pensez à la sélectionner dans le panneau des diapos avant de lancer le diaporama.

Pendant la lecture du diaporama, vous pouvez utiliser les flèches du curseur gauche et droite pour naviguer dans le diaporama. La touche Echap permet de sortir.

Impress offre bien d'autres fonctions, et notamment des modèles très colorés (servez-vous de la catégorie **Pages maîtresses** dans le panneau *Tâches* de droite), des transitions pour animer le passage d'une diapositive à l'autre (également dans le panneau *Tâches*) ainsi qu'une série d'outils similaires à ceux de LibreOffice Draw. Ils permettent de créer des formes, des légendes et d'autres symboles (tout est disponible dans le menu en bas de fenêtre).

Créer une invitation avec Draw

LibreOffice Draw permet de concevoir des pages dans lesquelles vous créez des objets graphiques, ce qui permet notamment de créer des affiches et des invitations. Malgré le nom du programme, les outils de dessin (*draw*) sont très simples et sont plutôt destinés à créer des organigrammes et des graphiques d'entreprise. Cela dit, les enfants apprécieront la facilité avec laquelle ils peuvent poser des étoiles, des smileys et des phylactères en plus de leurs images insérées.

Servez-vous de la Figure 6.5 si vous voulez vous exercer à créer une invitation avec LibreOffice Draw :

1. **Démarrez Draw ou demandez la création d'un nouveau dessin depuis le menu Fichier de n'importe quel module de LibreOffice**.

2. **Servez-vous de la barre d'outils située en bas de fenêtre pour sélectionner un premier outil de dessin**. En amenant le pointeur au-dessus des boutons, vous voyez apparaître une bulle d'aide.

   Choisissez le smiley gris qui est présélectionné dans le bouton *Formes des symboles* afin de le sélectionner.

3. **Amenez le pointeur dans la page de travail et cliquez puis maintenez tout en déplaçant le pointeur vers le bas et à droite**.

Lorsque vous déplacez le pointeur, vous voyez le dessin augmenter de taille pour remplir l'espace entre le point de départ et la position actuelle. Dès que vous relâchez, le dessin apparaît. Vous pouvez tracer la forme n'importe où puis la déplacer et la redimensionner.

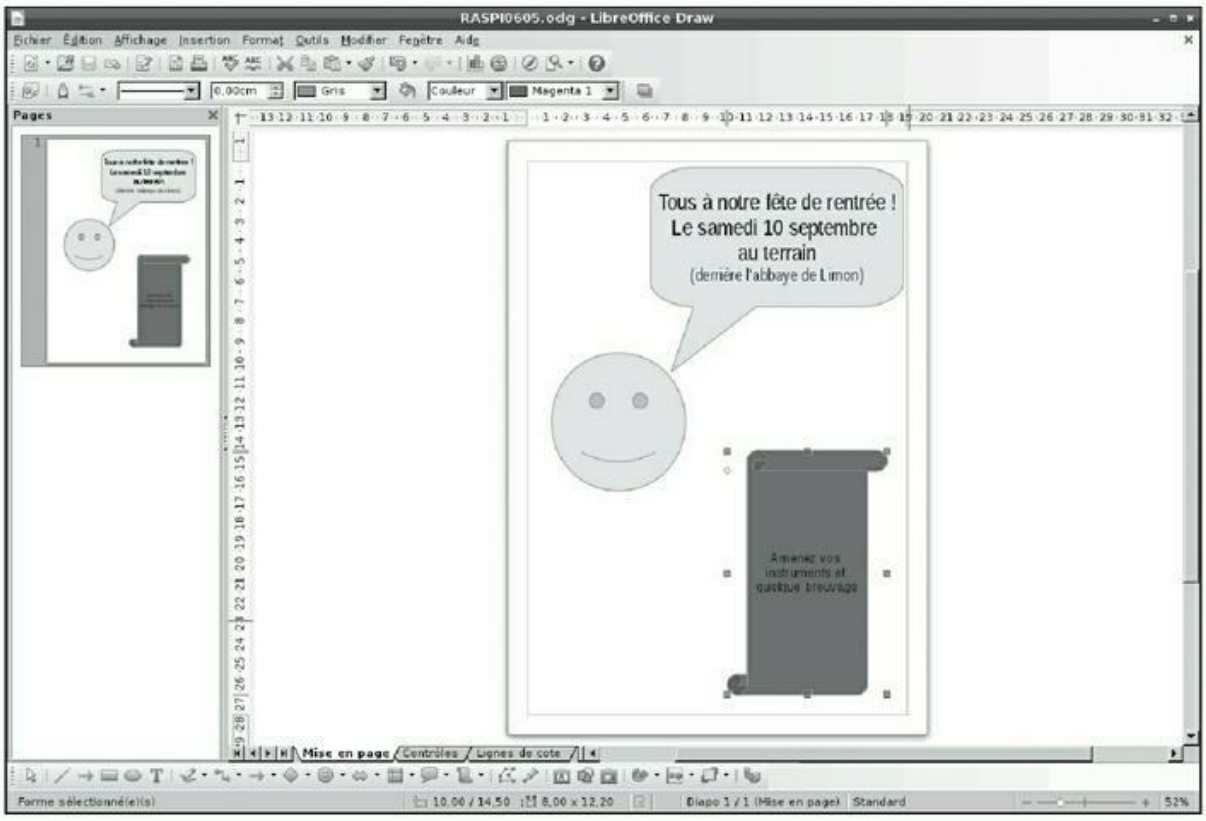


Figure 6.5 : Création d'une invitation avec LibreOffice Draw.

4. **Une fois le dessin en place, vous le repositionnez en cliquant puis en le faisant glisser**. Pour le redimensionner, cliquez puis servez-vous d'une des petites cases bleues qui apparaissent aux quatre coins.

5. **Pour ajouter un phylactère, servez-vous du groupe d'icônes *Légendes* dans la barre de menus inférieure**. (Cliquez l'icône puis cliquez dans la page en faisant glisser pour décider de la taille.)

Une fois la bulle mise en place, vous la redimensionnez et la repositionnez. Vous pouvez régler la position exacte de la flèche de bulle en vous servant du point jaune à son extrémité. Faites en sorte qu'elle vienne rejoindre la forme insérée auparavant.

6. **Cliquez dans la légende pour saisir votre texte**.

   Si vous saisissez trop de texte, il va déborder de la légende. Utilisez la touche Entrée pour passer à la ligne suivante si nécessaire, puis redimensionnez la forme pour que tout tienne.

7. **Certaines des icônes du bas sont dotées d'un menu déroulant auquel vous accédez *via* la petite flèche à droite de l'icône**. Ouvrez le menu du bouton **Étoiles** puis sélectionnez **Parchemin vertical** et placez un tel parchemin dans la page. Ajoutez du texte comme vous avez fait pour le phylactère.

8. **Vous pouvez ensuite personnaliser la couleur de chacun des trois éléments présents sur la page en sélectionnant l'élément puis en utilisant les options de la barre de menus dans la catégorie Style, en haut de fenêtre**.

   Notez que deux options de couleur sont proposées. Celle de gauche correspond à la couleur de ligne et celle de droite à la couleur de remplissage. Cliquez dans la zone qui affiche **Couleur**. Vous pouvez alors sélectionner un dégradé, un hachuré ou un bitmap (un motif coloré) plutôt qu'une couleur unie.

9. **Pour personnaliser la couleur du texte dans la bulle ou dans le parchemin, sélectionnez le texte en cliquant puis utilisez Ctrl + A pour être sûr d'avoir tout sélectionné**. Servez-vous alors de l'option **Couleur de police** tout à fait à droite de la barre des styles en haut de fenêtre.

10. **La même barre permet de modifier la police et la taille des caractères du texte**.

Vous devinez que bien d'autres choses sont possibles dans Draw. L'option **Texte** (une icône contenant un T dans la barre du bas) permet de placer du texte libre n'importe où, ce qui est très utile pour créer des affiches. L'option **Courbe** sert à tracer des lignes à main levée qui sont lissées automatiquement. L'option **Galerie Fontwork** sert à insérer un texte auquel a été appliqué une des déformations proposées. Lorsque vous avez inséré votre objet Fontwork, il ne reste plus qu'à cliquer dans le texte Fontwork initial pour saisir ce que vous voulez voir afficher avec cette déformation. Vous pouvez enfin ajouter vos propres images en utilisant le bouton **À partir d'un fichier** (menu du bas). Dès que l'image est chargée, vous pouvez la redimensionner et la repositionner pour bien l'adapter à votre mise en page.

Chapitre 7

Retouches photos avec GIMP

DANS CE CHAPITRE :

- **Installer GIMP**
- **Découvrir la fenêtre de travail de GIMP**
- **Redimensionner, rogner, tourner et inverser une photo**
- **Corriger des couleurs et nettoyer**
- **Convertir des images vers différents formats**

En ce début de XXIe siècle, nous vivons dans un monde plus documenté que jamais. Nous racontons tous les détails de notre vie sur des blogs et sur Facebook, et nombre d'entre nous portent toujours sur eux de quoi prendre des photos, que ce soit avec un smartphone ou une tablette numérique. Les mordus de photographie emportent même un véritable APN (appareil photo numérique). Quel que soit votre équipement et quoi que vous fassiez avec au quotidien, prendre des photos constitue un excellent moyen de dévoiler votre existence et de vous exprimer de façon créative.

Le petit Raspberry Pi peut jouer un rôle dans ce genre d'activités puisque vous pouvez vous en servir pour améliorer la composition et la qualité de vos photos. Sachez cependant que les photos des APN actuels sont assez volumineuses. L'ancien circuit Raspberry Pi qui n'avait que 256 Mo de mémoire avait bien du mal à les traiter, et arrivait même à se bloquer assez souvent. La version 512 Mo (qui correspond au modèle B vendu depuis le 15 octobre 2012) permet d'espérer des performances bien supérieures, mais il reste néanmoins à se montrer patient pour certaines opérations lourdes, ainsi qu'au premier démarrage. Les performances deviennent franchement correctes sur les modèles Raspberry Pi 2 ou 3, et vous pouvez envisager des traitements d'image sophistiqués.

Ce chapitre propose de présenter GIMP, qui est l'un des outils de traitement graphique et de retouches d'image les plus répandus. Vous y

découvrirez quelques astuces pour retoucher vos photos en les redimensionnant, rognant, tournant et inversant. Nous verrons ensuite comment corriger les couleurs et procéder à quelques retouches de nettoyage pour supprimer par exemple un détail indésirable dans un cliché.

Certains des savoir-faire de ce chapitre vous seront utiles dans d'autres projets du livre, et notamment le redimensionnement d'image pour en produire des variantes plus compactes, ce qui est indispensable pour les réutiliser par exemple sur un site Web (voir Chapitre 8) ou pour les insérer dans un programme de présentation comme LibreOffice Impress (voir Chapitre 6).

Installer et démarrer GIMP

Le programme que nous allons découvrir se nomme *GNU Image Manipulation Program*, ce qui correspond au sigle GIMP. C'est un outil gratuit, libre et très sophistiqué qui est utilisable aussi bien sous Linux que sous Windows et sous macOS. Voici l'unique commande à saisir dans un terminal pour demander l'installation de GIMP sur votre Raspberry Pi :

```
sudo apt-get update
sudo apt-get install gimp
```

À la moindre difficulté, reportez-vous au Chapitre 5 qui donne des détails concernant l'installation des logiciels.

Une fois l'installation achevée, vous devriez trouver la commande GIMP dans le sous-menu **Graphiques** du menu des Programmes de l'interface utilisateur graphique (voir Chapitre 4).

La fenêtre de travail de GIMP

La Figure 7.1 montre l'aspect général de la fenêtre de travail de GIMP. Au départ, la zone centrale est vide, hormis une vue estompée de la mascotte de GIMP, Wilber. Dans cette figure, nous nous sommes servis du menu **Fichier** pour charger une image qui est donc visible dans la zone de travail centrale.

À l'origine, GIMP avait été conçu sous forme de plusieurs fenêtres indépendantes, mais cela décontenançait beaucoup les personnes habituées aux autres programmes de création graphique. Nous préférons faire regrouper toutes les sous-fenêtres dans les limites de la fenêtre

principale, ce qui est notamment utile lorsque l'écran est de petite taille. Si ce que vous voyez sur votre écran est différent du contenu de la Figure 7.1, ouvrez le menu **Fenêtres** en haut de fenêtre principale pour choisir l'option **Mode fenêtre unique**.

Si vous ne voyez pas les deux volets gauche et droit comme dans la figure, ouvrez le menu **Outils** puis **Nouvelle boîte à outils** puis le menu **Fenêtres** et le sous-menu **Fenêtres ancrables**. Choisissez **Options de l'outil**.

La barre de menus en haut de fenêtre donne un aperçu de ce que le programme vous promet : Fichier, Édition, Sélection, Affichage, Image, Calque, Couleurs, Outils, Filtres, Fenêtres et Aide. Vous pouvez visiter les différents menus pour constater l'impressionnante quantité de commandes et d'options. Vous y trouverez notamment les commandes les plus utilisées pour lesquelles existent également des icônes dans la barre d'outils.

Le panneau de gauche est constitué de deux volets : celui du haut contient les icônes des différents outils et celui du bas les options de l'outil actif. Amenez le pointeur au-dessus sans cliquer pour obtenir une aide rapide. Dès que vous activez un outil, les options du volet inférieur changent en conséquence. Par exemple, si vous sélectionnez la **Brosse**, les options de brosse sont proposées, et notamment le réglage de l'opacité et le type de brosse.

Le panneau droit comporte lui aussi deux volets. Celui du haut contient des onglets pour les Calques, les Canaux, les Chemins et l'Historique. Les deux sections Calques et Historique (vous le voyez dans la Figure 7.1) sont indispensables à tout nouvel utilisateur parce qu'elles permettent d'effectuer des retouches de photos en toute sécurité.

L'onglet **Historique** donne accès à une liste de toutes les actions appliquées. Vous pouvez ainsi revenir en arrière sur une action dont le résultat ne vous convient pas.

Les calques constituent une technique très précieuse pour ajouter des éléments à une image sans mettre en péril son état initial. Si vous voulez par exemple ajouter du texte à une image, vous insérez ce texte sur un nouveau calque posé au-dessus de celui contenant l'image. Si vous changez d'avis, il suffira de supprimer le calque. L'outil **Texte** (celui dont l'icône est un A) génère automatiquement un nouveau calque pour le texte que vous saisissez. Si vous sélectionnez les outils de tracé, pensez à créer un nouveau calque d'abord au moyen du bouton **Nouveau calque** sous le volet (voir la Figure 7.1). Tout nouveau calque est posé *a priori* au-dessus des précédents, mais vous pouvez ensuite changer l'ordre des calques en intervertissant les lignes dans la liste du volet droit. Les calques du

premier plan sont ceux en haut de liste. Vous pouvez masquer un calque temporairement en cliquant dans le petit œil de sa ligne.

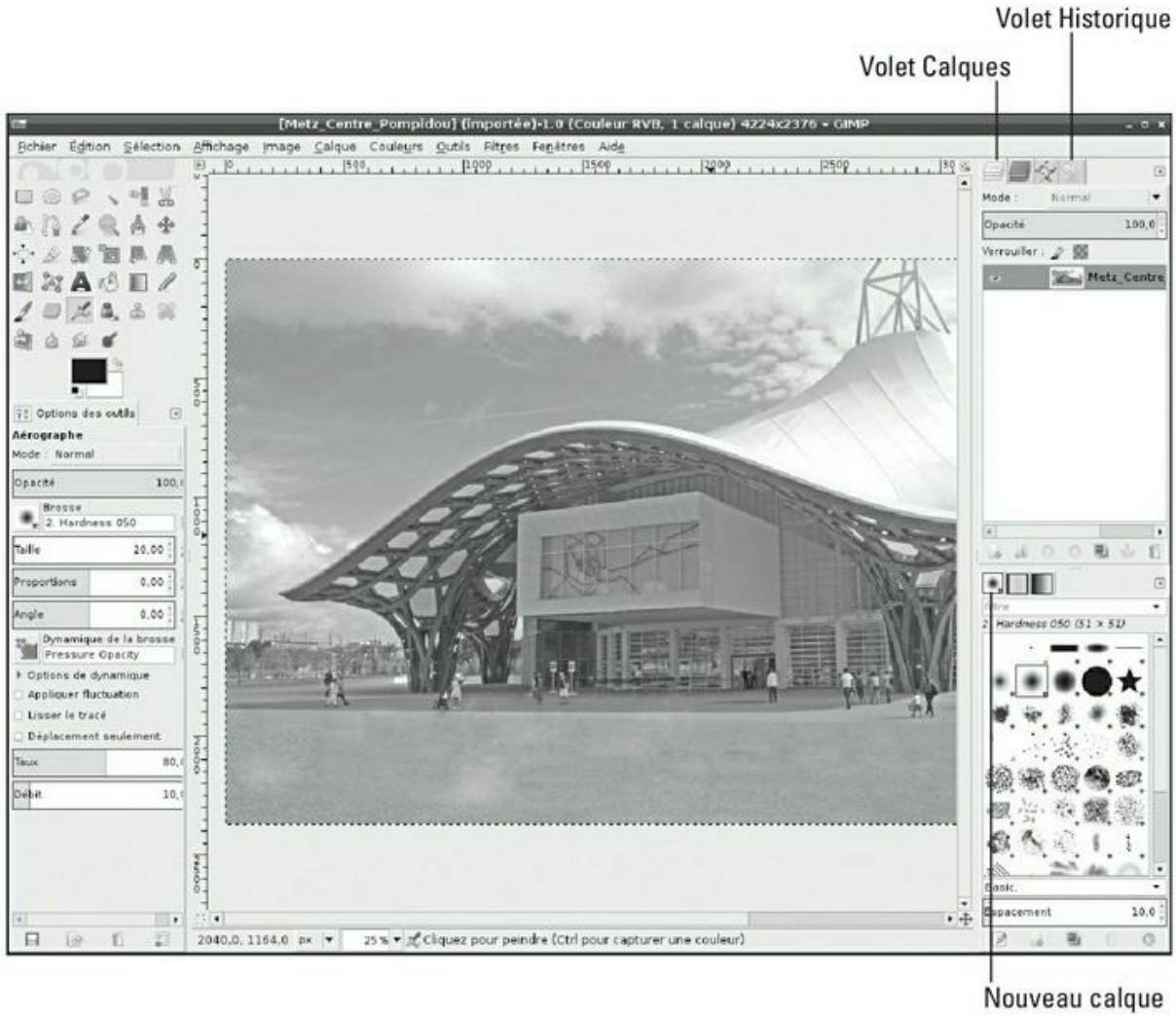

Volet Historique
Volet Calques
[Metz_Centre_Pompidou] (importée)-1.0 (Couleur RVB, 1 calque) 4224x2376 - GIMP
Fichier Édition Sélection Affichage Image Calque Couleurs Outils Filtres Fenêtres Aide
Nouveau calque

Figure 7.1 : Vue générale de la fenêtre de travail de GIMP en mode fenêtre unique.

La partie inférieure du panneau droit concerne les Brosses, les Motifs et les Dégradés. Les Brosses servent à dessiner ou à peindre sur l'image. Quant aux Motifs et aux Dégradés, ils servent en combinaison avec l'outil de remplissage qui sert à peindre d'un geste toute une partie de l'image en utilisant une couleur unie ou un motif. Nous ne présenterons pas ces différents outils de dessin dans ce bref chapitre, d'autant que leur utilisation sur le Raspberry Pi suppose une certaine patience, ce qui rend difficile le travail de grande précision et perturbe l'interactivité.

Vous pouvez retoucher la largeur des deux panneaux gauche et droit, ce que nous avons fait dans la Figure 7.1. Il suffit d'amener le pointeur sur un bord séparateur de panneau du côté de la zone centrale et de glisser puis déplacer dès que le pointeur a pris l'aspect d'une double flèche.

Redimensionner une image

Une des premières opérations que vous pouvez apprendre à réaliser avec votre Raspberry Pi consiste à redimensionner une image. Toutes les photos numériques sont constituées d'un réseau de pixels, qui sont des petits points colorés formant une mosaïque. Mon APN offre une résolution maximale de 4 272 pixels de large sur 2 848 pixels de haut. Il permet d'obtenir des clichés de haute qualité, donnant un bon résultat à l'impression, mais cette qualité est largement superflue pour des images qui ne seront affichées que sur un écran. En effet, qui dit haute résolution dit fichier volumineux, et un gros fichier va sensiblement ralentir votre Raspberry.

Vous pourrez donc fréquemment produire une version en basse qualité sans que la différence puisse se voir, à condition que cette variante ne soit destinée qu'à l'affichage vidéo.

Voici comment redimensionner une image avec GIMP :

1. **Ouvrez le menu Image en haut de fenêtre et choisissez Échelle et taille de l'image**.

   Vous voyez apparaître une boîte comme dans la Figure 7.2.

2. **Cliquez dans la zone de saisie Largeur et saisissez la largeur en pixels de l'image puis validez par Entrée**.

   Si votre but est de publier une photo de vacances sur votre site Web (nous verrons cela dans le Chapitre 8), il est inutile de préparer des images de plus de 500 pixels de large. Une largeur supérieure ne pourra pas être présentée dans le format classique d'une page Web et demandera un temps de chargement plus long. Si l'écran ne peut afficher que 1 024 pixels en largeur, donnez-vous comme limite maximale de largeur 800 pixels.

Dès que vous avez saisi une nouvelle valeur dans le champ Largeur puis avez frappé la touche Tab pour valider la saisie, cela provoque un recalcul automatique de la valeur dans le champ Hauteur pour que les proportions de l'image soient maintenues.

Vous pouvez donc plutôt saisir une valeur de hauteur, ce qui recalculera la largeur. Si vous êtes prêt à déformer l'image et donc à saisir manuellement deux valeurs pour la largeur et la hauteur, il faut cliquer dans le petit cadenas à droite de ces deux champs pour les désolidariser.

3. **Au lieu de spécifier des valeurs absolues pour la largeur et/ou la hauteur, vous pouvez demander un redimensionnement en pourcentage**. Commencez par ouvrir la liste déroulante qui affiche **px** à droite des deux champs et choisissez **percent**.

   Les valeurs affichées dans les champs de largeur et de hauteur deviennent des pourcentages. Vous pouvez par exemple saisir **50** pour réduire les deux dimensions de moitié. (Le poids du fichier sera divisé par quatre, puisque c'est une surface.) La taille de l'image en pixels est rappelée sous le champ **Hauteur**.

4. **Une fois que votre taille est bien spécifiée, pensez à cliquer le bouton Échelle en bas**.

Sous la zone d'édition de l'image en bas d'écran, la barre d'état rappelle les principaux paramètres du fichier, et notamment le taux de zoom actuel du côté droit. Si vous le réglez à 100 %, vous pouvez juger de la finesse des détails de l'image, et cela donne souvent plus de confort pour les retouches.

Figure 7.2 : La boîte des options de retaille d'image de GIMP.

Le fait de retailler une image réduit sa qualité. Cette perte sera clairement visible si vous comptez plus tard créer une impression de cette image. De ce fait, n'enregistrez pas l'image retouchée dans le fichier d'origine. Utilisez dans le menu **Fichier** la commande **Enregistrer sous** et spécifiez un autre nom de fichier.

Rogner une photo

Si votre photo comporte trop d'espace inutile sur les bords ou si vous voulez en changer la composition, vous pouvez faire une découpe, ce qui s'appelle rogner. Voici comment procéder :

1. **Sélectionnez l'outil dont l'icône ressemblant à un cutter (vous pouvez aussi utiliser le raccourci clavier Maj + C)**.

2. **Maintenez un clic dans l'angle supérieur gauche de la zone d'image à conserver puis faites glisser vers la droite et en bas avant de relâcher**. Dès que vous relâchez le bouton, vous voyez apparaître un cadre de sélection (Figure 7.3).

Le contenu de ce cadre est la partie de l'image qui sera conservée. Si votre sélection ne vous convient pas, ne perdez pas de temps à la recommencer car vous pouvez ajuster le cadre.

3. **Cliquez dans un des angles et faites glisser pour modifier la taille ou les proportions du cadre**. Vous pouvez même déplacer les bords pour ajuster la largeur ou la hauteur.

4. **Vous pouvez même déplacer le cadre en cliquant puis en faisant glisser par son centre**.

5. **Pour confirmer le redimensionnement, cliquez dans le cadre ou validez par la touche Entrée**.

Si vous vous trompez, vous disposez du raccourci habituel Ctrl + Z. Vous pouvez aussi utiliser le volet Historique (revoyez la Figure 7.1) qui permet de revenir en arrière dans les actions appliquées à l'image.

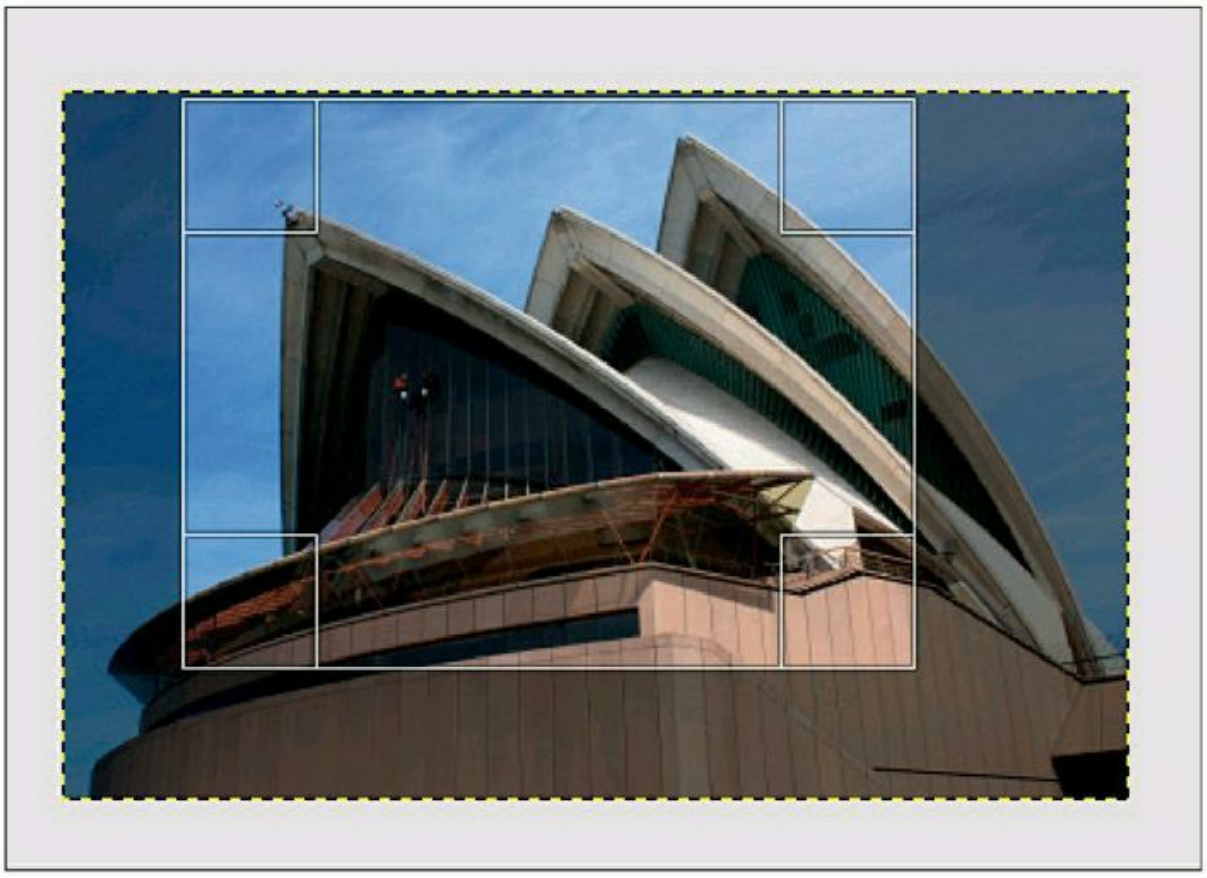


Figure 7.3 : Rognage d'une image dans GIMP.

Rotation et miroir

Si vous avez tourné votre APN pour prendre un cliché, vous aurez peut-être besoin de faire pivoter l'image. Le moyen le plus simple consiste à ouvrir le menu **Image** puis à choisir dans le sous-menu **Transformer**.

Vous pouvez faire tourner l'image dans le sens horaire ou antihoraire de 90°, la renverser de 180° ou lui appliquer un effet de miroir dans les deux sens.

Les rotations les plus simples pourront être réalisées plus rapidement avec un outil léger tel que LXDE Image Viewer (vu dans le Chapitre 4).

Si votre photo est un peu de travers, GIMP permet de l'ajuster avec précision. Sélectionnez l'outil **Rotation** (Maj + R). Vous pouvez alors indiquer un angle ou utilisez le glisser-déplacer pour faire tourner l'image lentement. Vous modifiez le point de pivotement en cliquant dans le cercle au milieu de l'image puis en le faisant glisser.

Corriger les couleurs

Comme tous les programmes de retouche d'image, GIMP offre de nombreuses options pour corriger les couleurs d'une photo. Ces options sont toutes réunies dans le menu **Couleurs**. Si votre cliché souffre d'une teinte indésirable, ou si vous voulez en ajouter un peu, servez-vous de la commande **Balance des couleurs** et réglez la proportion de rouge, de vert et de bleu. La commande **Luminosité-contraste** permet de faire ressortir des détails trop obscurs ou d'augmenter l'impact global de l'image.

Le sous-menu **Auto** un peu plus bas dans le menu **Couleurs** propose des techniques d'ajustement automatiques. Six modes sont proposés, mais les résultats peuvent être étonnants. Essayez malgré tout puisque vous pouvez annuler avec Ctrl + Z. L'option **Normaliser** peut rapidement améliorer des images qui semblent trop délavées. L'option **Balance des blancs** peut corriger des clichés qui manquent de piqué dans les zones très blanches et très noires.

Les options de couleur ne s'appliquent qu'au calque actif. Si vous avez ajouté d'autres calques à l'image, cliquez le calque **Photo** dans le panneau supérieur de droite avant de retoucher les couleurs

Corriger les imperfections

Lors de ses vacances en Australie, l'auteur avait pris un cliché d'une superbe plage immaculée à Darwin. Le résultat était superbe : un arbre isolé au premier plan, une mer à peine troublée et quelques lambeaux de nuages blancs dans un ciel bleu clair. En rentrant chez lui, il lui a fallu constater qu'un idiot avait laissé une canette de bière écrasée au premier plan.

Heureusement, GIMP propose un outil bien pratique pour faire disparaître ce genre de petit détail : l'outil **Clonage**. Il vous permet de réutiliser une zone de l'image comme motif à appliquer sur une autre zone de l'image. Dans notre exemple, nous partons d'un endroit propre de la plage pour faire disparaître la canette. En un clin d'œil, elle n'a jamais existé.

Voici comment utiliser l'outil de Clonage :

1. **Utilisez le curseur de zoom sous l'image pour vous rapprocher du défaut**. Servez-vous de l'ascenseur et de la barre de défilement horizontal pour bien vous positionner sur l'imperfection.

2. **Sélectionnez l'outil de Clonage dont l'icône ressemble à un tampon encreur (le raccourci est la lettre C)**.

3. **Amenez le pointeur sur une zone de l'image qui doit servir de source de la copie ; c'est la source du clonage**. Choisissez un endroit assez uni, ressemblant plus à une texture qu'à un motif et sans aucun détail remarquable. Un bout de ciel, de pelouse ou une plage convient parfaitement. Maintenez la touche Ctrl enfoncée et cliquez.

   La zone source reste marquée par un cercle avec un petit réticule.

4. **Dans les options de l'outil actif, dans le bas du volet gauche, vous pouvez voir le diamètre de la brosse en vigueur**. Choisissez un autre

diamètre ou une autre forme de brosse si nécessaire. Par défaut, c'est une brosse circulaire.

Les meilleurs résultats sont rendus par une brosse à contour progressif. Pour changer la taille de la brosse, vous disposez d'une option et d'un champ de saisie. Plus la taille de la brosse est grande, plus le motif va couvrir de surface à chaque geste. Amenez le pointeur sur le défaut et cliquez. Vous copiez ainsi un exemplaire de la zone source à l'endroit désiré.

5. **Si vous avez bien agi, le défaut devrait déjà s'estomper**.

   Si le motif que vous avez déposé semble contenir des détails indésirables, vous devez soit réduire le diamètre de la brosse, soit sélectionner une autre zone source. Répétez ce clonage jusqu'à faire disparaître totalement le défaut.

6. **Utilisez le curseur en bas de fenêtre pour ramener le taux d'agrandissement de l'image à 100 %**.

Vérifiez que vous ne voyez plus votre retouche. Si la correction du défaut se voit, essayez d'utiliser une autre zone source ou une autre taille de brosse.

Exporter une image dans différents formats

Il existe de nombreux formats de fichiers pour le stockage des images, mais tous les programmes ne savent pas lire tous les formats. Si vous préparez des images pour votre site Web, vous choisirez le format .jpg qui est le plus approprié pour les photos ou bien le format .gif préférable pour les illustrations.

Le format par défaut de GIMP se nomme .xcf (c'est son format natif). C'est un format sans perte qui stocke également des informations au sujet de la session de retouche et qui préserve tous les calques. Ce format n'a cependant pas été adopté par beaucoup d'autres logiciels.

GIMP vous permet d'exporter vos images dans un des formats les plus utilisés ou même de convertir une image d'un de ces formats dans un autre. Chargez d'abord l'image par le menu **Fichier** puis dans le même menu, utilisez la commande **Exporter**. La boîte qui apparaît ressemble beaucoup à la boîte d'enregistrement, sauf que vous pouvez choisir votre format en accédant aux détails de la rubrique **Sélectionner le type de fichier** (selon l'extension). Il ne reste plus qu'à choisir le format désiré puis à valider.

Les opérations de conversion sont gourmandes en mémoire. Si vous n'avez pas besoin de la plus haute qualité, pensez à rogner ou redimensionner vos clichés avant de les convertir.

Pour en savoir plus au sujet de GIMP

GIMP possède une très vaste panoplie de possibilités. Les fonctions sont toutes décrites dans la documentation accessible sur le site Web. Pour la consulter, ouvrez le menu **Aide** puis le sous-menu **GIMP en-ligne** et choisissez enfin **Site Web du Manuel utilisateur**. Vos pouvez aussi depuis votre navigateur accéder à la page http://docs.gimp.org/2.8/fr/.

Chapitre 8

Le Raspberry Pi comme centre multimédia

DANS CE CHAPITRE :

- **La distribution LibreELEC pour créer un centre multimédia**
- **Jouer des fichiers audio et vidéo d'une clé USB ou d'un réseau**
- **Visionner les photos**
- **Le multimédia dans Raspbian**

Dans ce chapitre, nous allons voir comment transformer une carte Raspberry Pi en un véritable centre multimédia capable d'exploiter des fichiers vidéo en haute définition et des fichiers audio.

Ce centre multimédia est fondé sur une distribution dédiée à cet usage qui va totalement remplacer celle que vous avez installée pour découvrir les chapitres précédents, Raspbian. La distribution multimédia que nous allons découvrir se base sur un logiciel multimédia appelé Kodi, qui est utilisé dans certaines box Internet. Vous pourrez ainsi visionner toutes les vidéos et écouter tous les fichiers audio qui deviennent accessibles soit sur des clés USB, soit sur le réseau local. Dans une certaine mesure, vous allez également pouvoir visionner des flux de chaînes de télévision et de radio sur Internet.

Attention ! Si vous mettez en place cette distribution à partir de la même carte mémoire que celle qui contient Raspbian, vous allez effacer tous les fichiers que vous avez stockés au cours des chapitres précédents. Mieux vaut créer une seconde carte système.

Nous verrons en fin de chapitre comment écouter des fichiers audio et visionner des fichiers vidéo en conservant la distribution Raspbian que nous utilisons dans tous les autres chapitres de ce livre.

Préparation du centre multimédia

Votre Raspberry Pi est capable de jouer des fichiers vidéo en haute définition à 1080p, ce qui lui permet de prétendre à un rôle de centre multimédia très économique. Le logiciel NOOBS que vous avez installé sur la carte mémoire permet de choisir parmi deux distributions Linux dédiées à cette fonction, toutes deux basées sur Kodi. Il s'agit d'OSMC et de LibreELEC. Nous choisissons dans ce chapitre de présenter la variante LibreELEC, car elle utilise les apparences par défaut de Kodi et ressemble donc beaucoup plus à d'autres box Internet. Elle est également moins gourmande en ressources, et conviendra donc mieux à des modèles Raspberry Pi moins performants.

Vous pouvez choisir la variante OSMC, tout en suivant la présentation de ce chapitre. Dans ce cas, vous pouvez revenir à l'aspect Kodi par défaut en choisissant les paramètres à gauche puis **Interface** puis **Skin**. Choisissez alors à nouveau **Skin** dans le panneau de droite puis choisissez l'apparence Estuary. Vous pourrez revenir à l'aspect par défaut d'OSMC au moyen de l'icône qui représente un engrenage à gauche, puis en choisissant **Interface settings** puis **Skin**. Vous ressortez de n'importe quel menu par la touche Echap.

Pour disposer de votre système multimédia, vous devez commencer par créer une carte mémoire microSD sur laquelle vous implantez NOOBS Light, comme décrit dans le Chapitre 2. Depuis cette carte, il suffit ensuite de demander l'installation de LibreELEC (voyez également le Chapitre 3).

Lors de l'implantation de LibreELEC, vous êtes guidé pendant la définition des paramètres principaux, et notamment le nom réseau, la connexion réseau et les paramètres d'accès distant, ce dernier permettant d'accéder aux fichiers du centre multimédia depuis le réseau.

Découverte du centre multimédia

Le premier écran de Kodi est visible dans la Figure 8.1. LibreELEC utilise l'interface Kodi simplifiée qui est prévue pour être pilotée par une télécommande. Dans ce chapitre, je suppose que vous utilisez une souris, mais je montre comment utiliser une télécommande vers la fin du chapitre.

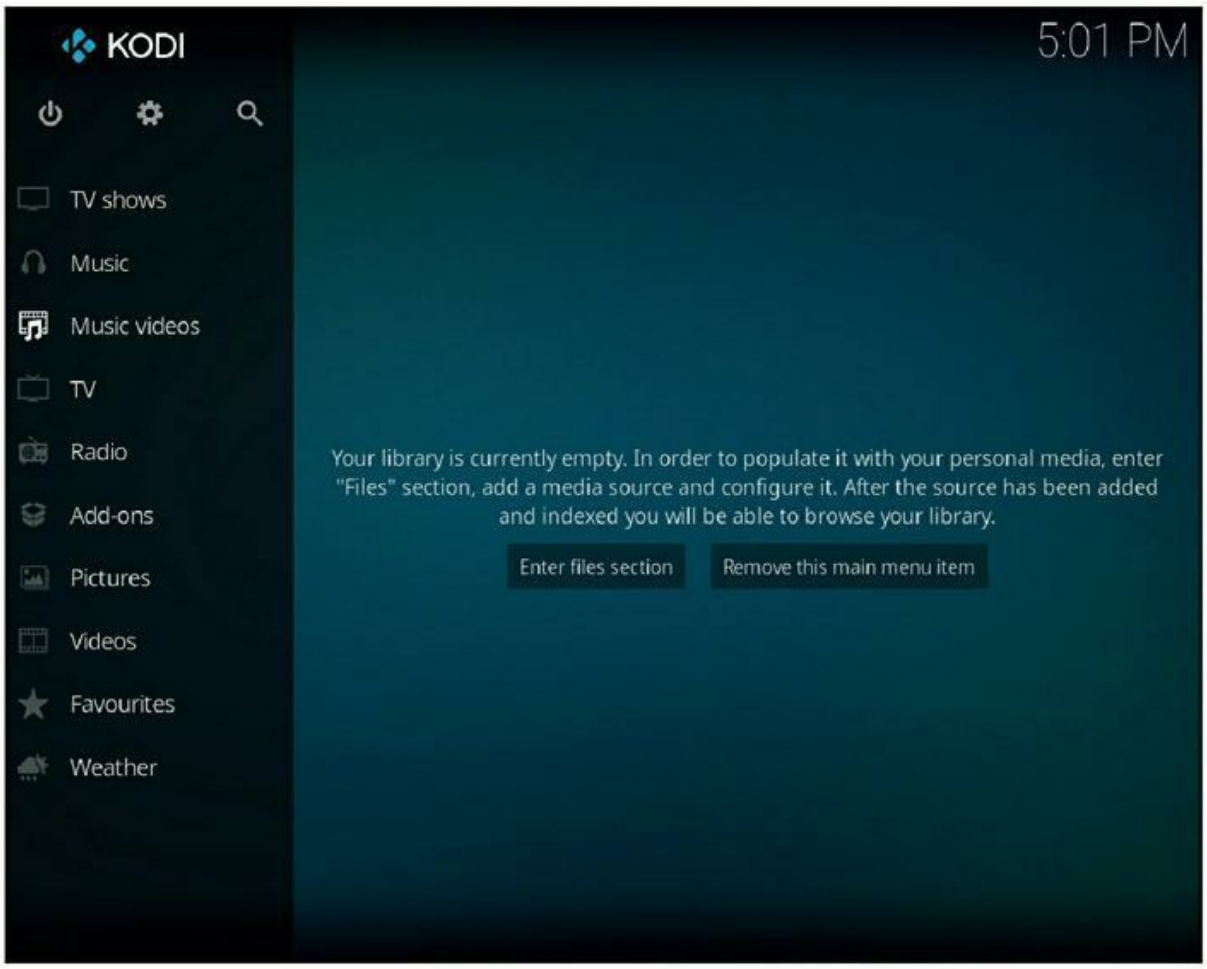

KODI
5:01 PM
TV shows
Music
Music videos
TV
Radio
Add-ons
Pictures
Videos
Favourites
Weather
Your library is currently empty. In order to populate it with your personal media, enter "Files" section, add a media source and configure it. After the source has been added and indexed you will be able to browse your library.
Enter files section
Remove this main menu item

Figure 8.1 : Le menu principal de LibreELEC avec la surface d'affichage sur la droite. Pour l'instant, aucun fichier n'a été ajouté.

Le menu qui constitue le panneau de gauche permet d'accéder aux différents types de fichiers multimédias (vidéos, émissions télévisées, musiques et clips vidéo). Il suffit d'amener le pointeur sur une des zones pour voir les options et sous-menus correspondants dans la partie de droite. Par exemple, la catégorie **Music** montre les catégories disponibles, les derniers albums écoutés, les derniers ajoutés, des albums au hasard, des artistes et albums encore non écoutés, et les favoris. Servez-vous des flèches du curseur ou de la molette de la souris pour naviguer parmi les options. Pour sortir d'un menu, utilisez la touche Echap.

Il suffit de cliquer pour lancer l'exécution d'un fichier. Pour faire une pause, utilisez la barre d'espace et la touche Echap pour revenir au menu.

Pour accéder aux émissions télévisées et aux chaînes radio, vous devez vous équiper d'un périphérique tuner et d'un décodeur TV. Pour tout détail, visitez la page suivante :

http://kodi.wiki/view/PVR_backend

Tout en haut à gauche, vous trouvez trois boutons très importants : le bouton pour éteindre votre Raspberry Pi, celui pour accéder aux paramètres (un engrenage) et la loupe pour chercher parmi les contenus.

Ajouter des fichiers multimédias

Avant de pouvoir profiter de vos fichiers, vous devez indiquer à Kodi où ils se trouvent. Voici les options disponibles :

- **USB drive (disque USB) :** Vous pouvez insérer une clé USB ou un disque directement dans un port USB de votre carte. Vous verrez apparaître un message en haut à droite pour confirmer que le périphérique USB a bien été connecté logiquement (monté) et que son contenu est accessible.
- **Networked media (médias réseau) :** Vous pouvez relier votre Raspberry à votre réseau domestique afin d'accéder à des supports de stockage. L'un des auteurs a ainsi relié son Raspberry à son PC sous Windows pour écouter ses musiques et visionner ses films depuis Kodi. Si vous disposez d'un routeur avec un serveur multimédia, il suffit de partager des fichiers qui se trouvent sur vos périphériques USB. En général, les appareils utilisent le standard uPNP (*Universal Plug and Play*). Kodi peut créer une librairie de tous vos fichiers multimédias puis générer un index pour permettre un accès rapide. Il faut d'abord ajouter les contenus à librairie.

Ajouter des fichiers audio

Voici comment ajouter des fichiers audio à votre librairie :

1. **Dans le menu de gauche, amenez le pointeur au-dessus de Music**.

   Si vous avez déjà ajouté de la musique, vous voyez apparaître des options pour naviguer par artiste, par album, par année, *etc*.

2. **Dans le menu des catégories en haut de l'écran, choisissez Files**. Servez-vous de la molette si nécessaire.

   Vous voyez réapparaître les dossiers qui existent déjà et les noms de tous les périphériques connectés.

3. **Choisissez l'option Add music**.

4. **Dans la boîte d'ajout d'une source de musique, choisissez le bouton de parcours Browse et naviguez jusqu'à trouver vos fichiers audio (Figure 8.2).**

   Si vous avez inséré une clé USB, vous la trouvez en cherchant *Root filesystem, media* puis le nom de la clé. Pour trouver des fichiers réseau, essayez les options **Windows Network SMB** ou **Network File System**.

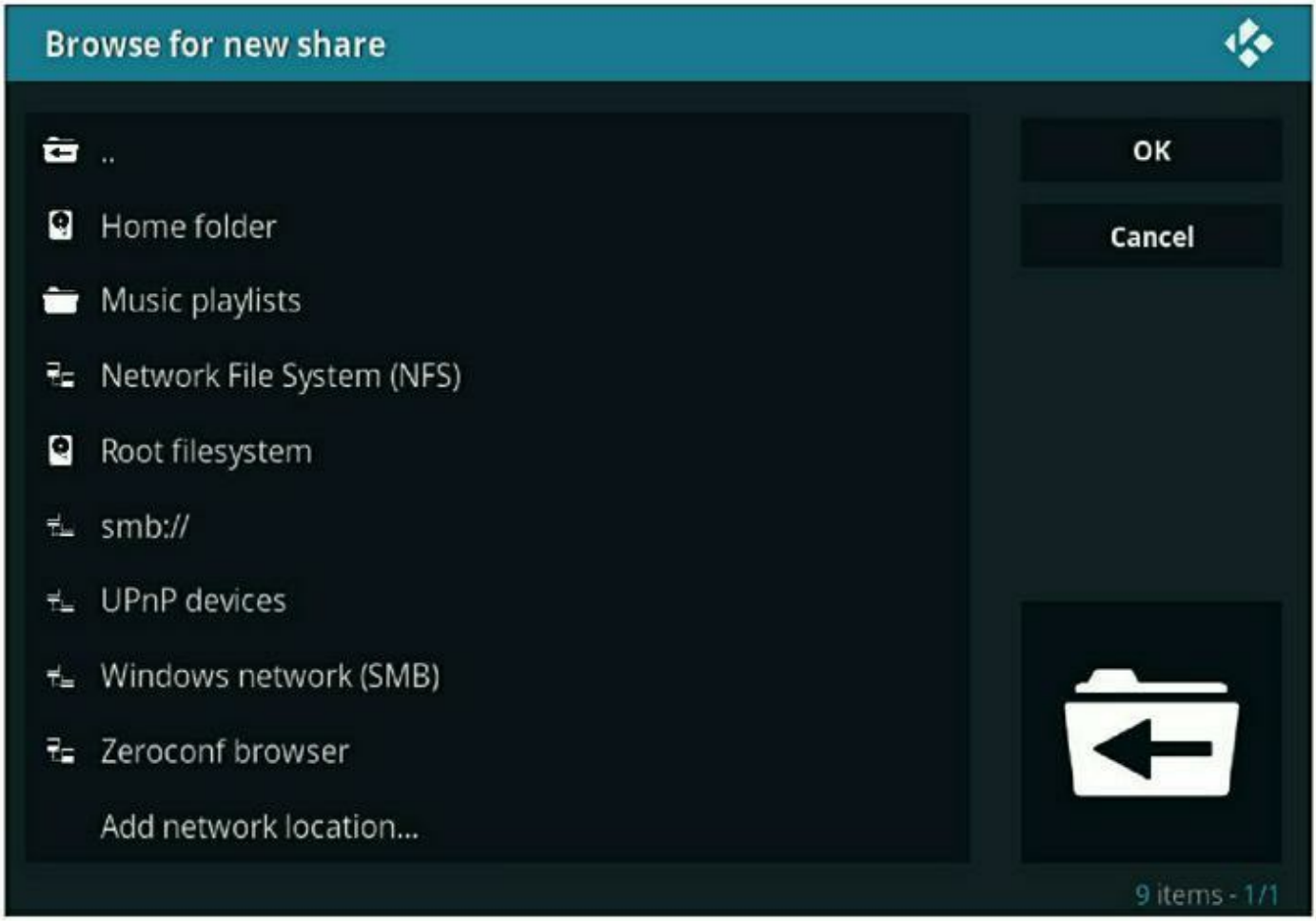


Figure 8.2 : Options de navigation pour ajouter des fichiers à Kodi.

Vous devrez sans doute indiquer votre nom d'utilisateur et votre mot de passe. Pensez à activer l'option pour que le système se souvienne de ces éléments afin de ne pas avoir besoin de les ressaisir à chaque fois. Cliquez dans un dossier pour y accéder. Le nom symbolique **..** au début permet de remonter d'un niveau parmi les dossiers.

5. **Cliquez OK une fois que vous avez atteint le dossier contenant les fichiers à ajouter**.

6. **Dans la boîte d'ajout d'une source, donnez un nom à cette source, si nécessaire**.

Avec un nom suggestif, vous aurez moins de mal à identifier les différentes sources dans les menus.

7. **Cliquez OK**.

8. **Quand un message vous le demande, confirmez que vous voulez ajouter les fichiers multimédias**.

Kodi n'accepte pas d'ajouter des contenus dont il ne reconnaît pas le format, et notamment les fichiers audio personnels. Vous pouvez cependant les visionner au moyen du navigateur. Accédez à gauche au menu **Music**, choisissez **Files** puis choisissez le dossier contenant les fichiers. Vous pouvez ensuite accéder aux sous-dossiers et choisir les fichiers audio que vous voulez jouer.

Ajouter des fichiers vidéo

Voici comment procéder pour ajouter des vidéos à une librairie :

1. **Dans le menu de gauche, choisissez Movies**.

2. **Si vous ne voyez pas cette option, c'est qu'elle est masquée**. Utilisez la molette de la souris pour voir les autres options du menu.

   Si vous avez déjà ajouté des films, vous pouvez en voir la liste au moyen du nom spécial **..** en début de liste. Cliquez deux fois ce nom puis choisissez **Files**. Vous allez ainsi voir les noms des dossiers que vous avez déjà ajoutés.

3. **Dans le bas de la liste des dossiers, choisissez Add Videos**.

4. **Choisissez le bouton de parcours Browse et naviguez jusqu'à trouver les fichiers**.

   Si vous avez inséré une clé USB, vous la trouvez en cherchant *Root filesystem, media* puis le nom de la clé. Pour trouver des fichiers réseau, essayez les

options **Windows Network SMB** ou **Network File System**. Cliquez dans un dossier pour y accéder. Le nom symbolique **..** au début permet de remonter d'un niveau parmi les dossiers.

5. **Dès que vous êtes dans le dossier désiré, cliquez OK**.

6. **Dans la boîte des options, cliquez la mention This directory contains puis choisissez Movies**.

7. **(Facultatif) Souvent, les fichiers vidéo sont stockés dans des sous-dossiers qui portent le même nom que le titre du film**. Si c'est le cas, activez l'option correspondante.

8. **Cliquez OK pour confirmer**.

   Si cela vous est demandé, confirmez que vous voulez réactualiser les éléments de ce chemin.

Kodi n'accepte pas d'ajouter des contenus dont il ne reconnaît pas le format, et notamment les fichiers vidéo personnels. Vous pouvez cependant les visionner au moyen du navigateur. Accédez à gauche au menu **Movies**, choisissez **Files** puis choisissez le dossier contenant les fichiers. Vous pouvez ensuite accéder aux sous-dossiers et choisir les fichiers que vous voulez lire.

Le processus est le même pour ajouter des émissions télévisées ou des clips vidéo. Il suffit de choisir la catégorie **TV Shows** ou **Music Videos** à l'Étape 5.

Ajouter des photos

Le processus est le même que pour ajouter des fichiers audio ou vidéo. Vous commencez par choisir la catégorie **Pictures** à gauche puis la commande **Add Pictures**.

Pour voir le contenu d'un dossier d'images, amenez le pointeur au-dessus de **Pictures** puis choisissez le nom du dossier. Vous naviguez parmi les photos avec les flèches gauche et droite. Les options en bas à gauche donnent accès à un menu permettant notamment d'afficher un diaporama.

Pour changer de dossier, ou pour ajouter de nouvelles images, utilisez le nom symbolique **..** en début de contenu de dossier.

Exploiter des flux multimédias

Un flux multimédia, appelé également *stream*, est une transmission en direct depuis une source vers votre Raspberry Pi *via* Internet. Pour profiter des flux, il faut donc une connexion Internet de bonne qualité.

Pour profiter d'un flux, il faut installer des extensions (*add-ons*) qui sont des applications permettant d'accéder à une source de flux. Une extension audio permet par exemple d'écouter des stations de radio Internet et d'accéder à des services de musique en ligne. Les extensions vidéo permettent de visionner des chaînes de télévision. La liste exacte des extensions disponibles varie au cours du temps, car des services apparaissent et d'autres disparaissent. Les extensions pour les flux de musique et les flux vidéo sont particulièrement instables car les diffuseurs préfèrent que ceux qui utilisent leurs services se servent de leurs logiciels et de leurs passerelles propriétaires.

Pour ajouter une extension, choisissez la catégorie **Add-ons** à gauche puis **Install from repository** en haut d'écran. Dans le menu qui apparaît, choisissez **Kodi Add-on Repository** pour chercher toutes les extensions disponibles dans le référentiel Kodi officiel. Tous ces modules complémentaires ont subi un minimum de tests.

En plus des flux de musiques et de vidéos, vous pouvez installer des extensions pour obtenir des informations de météo, des économiseurs d'écran (dans la catégorie **Look and Feel**) et des images. Si vous installez un économiseur d'écran, pensez à l'activer dans les paramètres d'interface.

Écouter de la musique

Pour écouter vos morceaux, accéder à la catégorie **Music** à gauche pour voir tous les fichiers que vous avez ajoutés. Vous pouvez naviguer par genre, artiste, album et morceau au moyen du menu des catégories en haut. Vous pouvez aussi choisir **Files** pour accéder directement à un dossier en particulier. D'autres options dans le panneau principal permettent de réécouter les derniers morceaux joués ou les albums récents, ou bien d'effectuer une lecture aléatoire. Pour lancer un morceau, il suffit de cliquer.

Pendant l'écoute, le bouton d'options en bas à gauche permet de contrôler la lecture. Pour afficher tous les paramètres du morceau courant en plein

écran, utilisez l'icône qui montre quatre flèches. Un économiseur d'écran est disponible, il est activé après quelques instants. Utilisez la touche Echap pour revenir au menu. Pour mettre en pause, utilisez la barre d'espace.

Vous pouvez naviguer parmi les morceaux pendant la lecture et ajouter des morceaux à la file d'attente en cliquant droit puis en choisissant **Queue Item**.

Dans la section des catégories du menu **Music**, vous disposez d'une option **Playlist** qui permet de créer des listes de lecture. Pour ajouter un morceau à la liste, cliquez droit dans son nom puis choisissez **Add**. Une fois la liste terminée, choisissez la commande **Save** dans le menu de gauche.

Vous pouvez même créer des listes intelligentes, en définissant au préalable leurs règles. Vous établissez ainsi une liste pour une année, pour un genre ou pour un artiste.

Une liste intelligente ne peut réunir que les morceaux présents dans la librairie. Il existe cependant des listes standard qui réunissent des morceaux provenant de n'importe lequel de vos appareils connectés.

Visionner des vidéos

Vous trouverez vos vidéos dans les quatre catégories **Movies**, **TV Shows**, **Music Videos** et **Videos**. Le contenu exact de chaque catégorie dépend bien sûr de ce que vous y avez ajouté et du type de fichiers implantés.

Le système Kodi sait lire les formats vidéo H.264 jusqu'à la résolution 1080p. Autrement dit, vous devez pouvoir visionner en haute définition la plupart des fichiers aux formats .mp4, .m4v et .avi. Sachez cependant que les fichiers qui sont protégés par des droits d'accès, notamment ceux achetés dans le magasin iTunes, ne peuvent pas être lus dans Kodi.

Pour visionner d'autres formats, il vous faudra acheter une licence sur le site Web officiel de Raspberry Pi (raspberrypi.org). Une licence pour les formats MPEG2 et VC1 est disponible pour quelques euros. Chaque licence n'est valable que pour une carte Raspberry Pi. Vous devrez ajouter le numéro de licence dans le fichier config.txt (décrit dans l'Annexe A). L'opération est plus simple si vous démarrez le système depuis une autre carte mémoire SD puis utilisez le lecteur de carte du Raspberry Pi pour modifier la carte contenant le système Kodi. Le fichier config. txt doit être stocké dans le dossier principal de la carte Kodi, et non dans le sous-dossier *boot* comme vous pourriez vous y attendre.

Pour lire un film, accédez à la catégorie **Movies** à fin de voir les différentes possibilités. Cliquez le nom d'un film pour en lancer la lecture. Vous pouvez obtenir la liste de tous les films existants dans la catégorie. Certains seront agrémentés de l'affiche et d'un synopsis s'ils ont été trouvés sur Internet. En haut du menu, utilisez l'entrée **..** deux fois pour accéder à des fichiers qui ne sont pas présents dans la librairie. Comme pour les fichiers audio, la barre d'espace permet de contrôler la lecture.

Visionner des photos

Choisissez **Pictures** dans le menu principal pour accéder à un navigateur de fichiers qui montre tous les dossiers d'images ajoutés. Le système reconnaît les formats les plus répandus, et notamment JPEG, BMP et TIFF. Il génère des vignettes pour les photos et pour les dossiers. Servez-vous des flèches gauche et droite pour circuler parmi les photos et de la touche Echap ou clic-droit pour revenir au menu.

Paramétrage de LibreELEC

Pour accéder aux paramètres de Kodi/LibreELEC, cliquez l'icône représentant un engrenage en haut à gauche. Voici les principales options :

- **Player settings :** Permet d'enchaîner automatiquement ou pas la lecture du morceau suivant, d'ajouter un fondu enchaîné et d'afficher les sous-titres.
- **Media settings :** Sert à gérer les sources de fichiers dans la librairie et les sites qui diffusent des informations sur les artistes et les albums.
- **Interface settings :** Permet de changer l'aspect de Kodi, la région et l'économiseur d'écran. C'est ici qu'il faut activer l'économiseur si vous en avez envie.
- **System settings :** Permet de régler la sortie audio, de configurer les notifications dans Kodi, de gérer l'alimentation et la connexion Internet.

- **LibreELEC :** Réunit les options de gestion de connexion Internet et de connexion sécurisée SSH pour l'accès distant.
- **File Manager :** Permet de copier des fichiers d'un dossier et d'un support de stockage à l'autre.

Par défaut, le son est transmis par le câble HDMI. Si vous voulez utiliser le câble HDMI pour l'image, mais récupérer le son sur la sortie audio du Raspberry Pi, il faut basculer ici l'option de **HDMI** vers **Analog**. L'option se trouve dans la catégorie **System Settings**. Dans le menu des paramètres, choisissez à gauche **Audio** puis **Audio output**.

Utiliser une télécommande

Les nombreuses possibilités multimédias présentées dans ce chapitre permettent réellement de se doter d'un véritable centre multimédia. Il ne lui manquerait qu'une chose : une télécommande.

Plusieurs possibilités s'offrent à vous pour contrôler votre centre à distance. Vous pouvez adopter un périphérique USB, une télécommande de récupération, un clavier sans fil, ou même une manette Xbox. Vous pouvez également utiliser l'application de contrôle à distance de Kodi qui est disponible pour iOS et pour Android afin de contrôler le centre depuis votre telliphone.

Les paramètres de la télécommande se trouvent dans le menu **System Settings**, catégorie **Input**. Cela dit, il est possible que la télécommande fonctionne sans avoir besoin de toucher à un quelconque paramètre. Faites d'abord des essais.

Si votre téléviseur supporte le standard HDMI CEC, il est intéressant de faire contrôler votre centre multimédia avec sa télécommande. Reliez votre Raspberry Pi à un connecteur HDMI du téléviseur. Vous devriez voir apparaître une nouvelle source nommée XMBC. Servez-vous de la télécommande du téléviseur pour choisir cette source et voir apparaître l'écran de votre centre multimédia sur le téléviseur !

Éteindre le centre multimédia

Nous conseillons de toujours éteindre correctement le centre multimédia au moyen de l'icône correspondante (dans le coin supérieur gauche de

l'écran principal de Kodi). Si vous coupez brutalement l'alimentation, vous prenez le risque d'altérer les données sur la carte mémoire.

Le multimédia dans Raspbian

Vous pouvez tout à fait jouir de vos fichiers multimédias sans passer par l'installation d'un centre multimédia dédié. Vous pouvez en effet écouter vos fichiers audio et regarder vos vidéos dans le système Raspbian. Démarrez donc ce système en installant la carte mémoire correspondante. Si vous partez de NOOBS, choisissez d'installer Raspbian pour démarrer.

Un lecteur multimédia très répandu, et qui fonctionne dans l'environnement PIXEL de Raspbian, est le lecteur VLC Media Player. Commencez par l'installer depuis la ligne de commande de la façon suivante (voir aussi le Chapitre 5) :

```
sudo apt-get install vlc
```

Une fois que l'outil est installé, vous le trouverez dans la catégorie **Son et vidéo** ou **Accessoires**. Cliquez pour démarrer.

Le menu **Média** de VLC permet d'ouvrir un fichier ou un répertoire, contenant par exemple un album entier. Lorsque c'est possible, VLC affiche la pochette de l'album, et vous pouvez ouvrir le menu **Vue/View** puis **Liste de lecture/Playlist** pour accéder à la liste des morceaux (Figure 8.3). Le panneau de gauche permet de choisir parmi les différentes sources possibles, y compris des flux Internet. Choisissez à nouveau **Liste de lecture** dans le menu **Vue** pour revenir à l'affichage de la pochette.

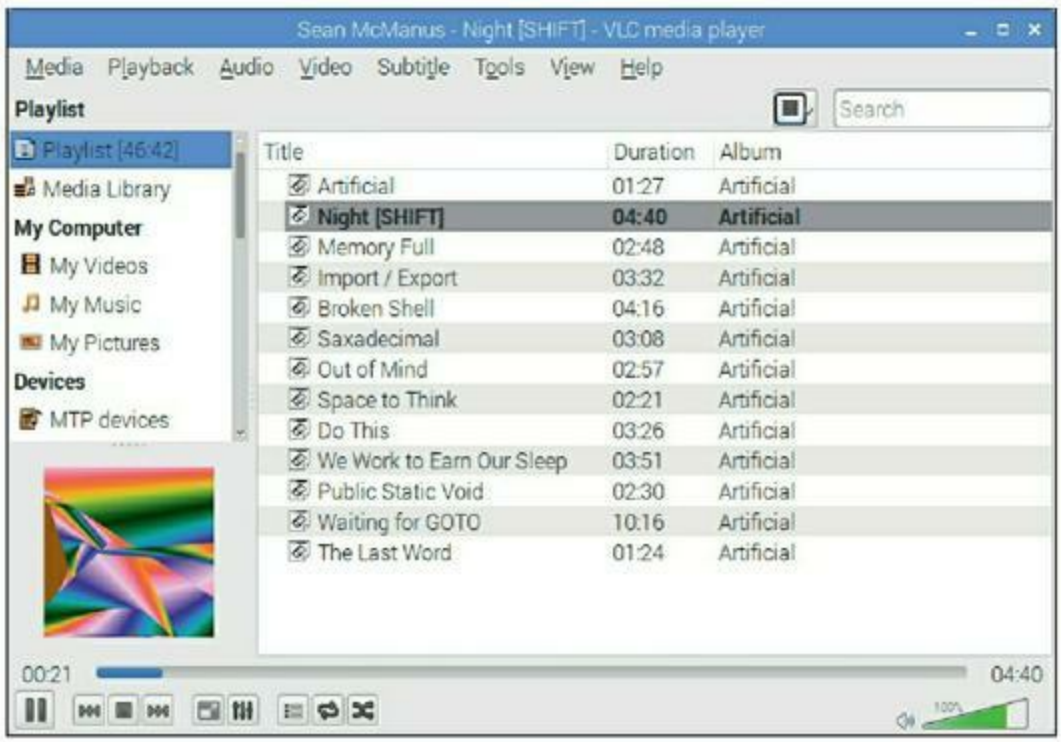


Figure 8.3 : Lecture audio dans VLC Media Player.

Le bas de la fenêtre réunit les commandes de navigation parmi les morceaux et le contrôle du volume est en bas à droite.

PARTIE 4
Apprendre à programmer avec le Raspberry Pi

DANS CETTE PARTIE :

- Découverte de l'interface de l'atelier Scratch et mise en pratique par la création d'une animation.
- Utilisation de Scratch pour créer un programme de jeu d'arcade simple et personnalisable avec vos dessins.
- Découverte du langage Python et création d'un générateur de table de multiplication et d'un Chatbot ou simulateur d'intelligence artificielle.
- Utilisation de la librairie Pygame Zero pour créer un jeu d'arcade que vous personnaliserez avec vos sons et vos images.
- Découverte des techniques pour créer automatiquement un bâtiment dans Minecraft avec des instructions en Python.
- Premiers pas dans la création de musique électronique avec le logiciel Sonic Pi.

Chapitre 9

Introduction à la programmation visuelle avec Scratch

DANS CE CHAPITRE :

- **Démarrage de Scratch**
- **La fenêtre de l'atelier Scratch**
- **Placement et redimensionnement de vos objets (*sprites*)**
- **Déplacement de vos objets**
- **Modification de l'apparence de vos objets**
- **Ajout de bruits et de musique**

Une des raisons d'être du Raspberry Pi est de proposer un tremplin à la prochaine génération de programmeurs. Le langage Scratch constitue une voie d'accès idéale à ce tremplin. Scratch vous permet de concevoir et de réaliser aisément vos animations et vos premiers jeux vidéo tout en découvrant quelques-uns des concepts que doit maîtriser tout programmeur professionnel au quotidien.

Scratch a d'abord été conçu pour être abordable à tous les âges. Grâce à son interface utilisateur ludique, vous savez à tout moment où vous en êtes et ce que vous pouvez faire sans devoir mémoriser des commandes, et vous obtenez vite des résultats intéressants. Scratch est en effet accompagné d'une collection d'objets graphiques et d'effets sonores, ce qui vous permet de produire votre premier programme Scratch en quelques minutes.

Nous vous proposons dans ce chapitre de découvrir l'atelier Scratch en vous guidant dans vos premiers pas. Le chapitre suivant profitera de cette

initiation pour vous montrer comment réaliser un jeu d'arcade simple mais complet.

Programmer, c'est quoi ?

Avant de plonger dans Scratch, éclaircissons quelques termes du jargon typique de cet univers. Un *programme* est une série d'ordres (des instructions) que l'unité centrale d'un ordinateur va exécuter en séquence pour produire un résultat visuel, sonore ou numérique, comme l'animation d'un personnage de jeu à l'écran. La séquence d'instructions peut être très longue et complexe, car elle doit décrire de façon très élémentaire ce que l'ordinateur doit réaliser. Faire rebondir un cercle symbolisant une balle sur le bas de l'écran suppose d'écrire des instructions pour tracer un cercle, pour le déplacer puis d'y ajouter des tests de collision avec le sol pour déclencher le rebond en calculant son angle.

La *programmation* est l'art de concevoir et de réaliser des programmes informatiques, c'est-à-dire des logiciels. De nombreux outils et langages sont disponibles pour créer des programmes ; Scratch est l'un d'eux. Dans le Chapitre 12, vous découvrirez un autre langage nommé Python.

Scratch et Python sont tous deux des *langages de programmation*, deux manières différentes de rédiger le *code source* qui doit ensuite être converti en langage machine exécutable par l'ordinateur. Chaque langage a des domaines de prédilection. Scratch est parfait pour s'initier et créer des jeux simples, mais il ne permet pas de créer un logiciel de traitement de texte, ni de faire des calculs numériques complexes. Le langage Python demande plus de temps d'apprentissage, même pour créer un jeu, mais il est bien plus puissant et polyvalent que Scratch, puisqu'il est utilisé dans les environnements professionnels.

Scratch 1 ou Scratch 2 ?

Dès le départ, votre système Raspbian propose deux générations de l'atelier de création Scratch :

- **Scratch 1 :** C'est la mouture originale de Scratch, qui en est actuellement à la version Scratch 1.4.
- **Scratch 2 :** Cette génération, plus récente, est le fruit d'une décision des concepteurs de profiter d'un mécanisme de gestion graphique nommé

> **Flash**. C'est la version 2 que l'on utilise quand on accède à Scratch dans sa version en ligne, sans installation (`https://scratch.mit.edu`). Scratch 2 ajoute quelques nouveautés à son prédécesseur : la possibilité pour un lutin de se copier et celle de créer de nouveaux blocs. (Nous revenons sur ces nouveautés en toute fin de chapitre.) Contrairement aux habitudes, la version la plus récente n'est pas toujours la meilleure. En effet, Scratch 2 oblige à installer Flash, qui consomme de la puissance, ce qui limite Scratch 2 aux plus récents circuits Raspberry Pi 2 et 3. Certains programmes, notamment des jeux vidéo, sont sérieusement ralentis dans cette version de Scratch, à cause des couches de traitement imposées par Flash. Si vous avez besoin de performances pour un jeu vidéo, vous aurez peut-être intérêt à conserver Scratch 1.

Les deux versions de Scratch ont été enrichies pour mieux contrôler des périphériques électroniques depuis un Raspberry Pi, comme nous le montrerons dans le Chapitre 16.

Si vous découvrez Scratch, adoptez le Scratch original, afin de ne pas perdre en performances. Si vous avez pris l'habitude de Scratch 2 Online, ou si vous avez absolument besoin des quelques nouveautés de Scratch 2, passez à cette version.

Dans nos exemples, nous allons utiliser la version originale Scratch 1, d'autant que Scratch 2 ne fonctionne pas sur les cartes Raspberry plus anciennes comme la Model B+.

Le code source de vos programmes écrits pour Scratch 1 est directement utilisable dans Scratch 2, mais pas l'inverse.

Pour démarrer Scratch, ouvrez le menu général puis le menu **Programmation**. Vous trouvez alors les deux variantes Scratch et Scratch 2. Choisissez **Scratch** pour que vos affichages correspondent aux images de ce livre.

La fenêtre de l'atelier Scratch

La fenêtre Scratch est structurée en quatre volets (voir Figures 9.1 et 9.2). La scène (*stage*) est la zone libre en haut à droite dans Scratch 1 (en haut à gauche dans Scratch 2) ; c'est ici que le programme va s'exécuter. Elle contient déjà un chat immobile au milieu ; il est prêt à obéir à vos premières instructions, comme nous allons le voir très bientôt.

Si les menus apparaissent en anglais, cliquez le petit globe terrestre dans la barre de menus et choisissez **Français**.

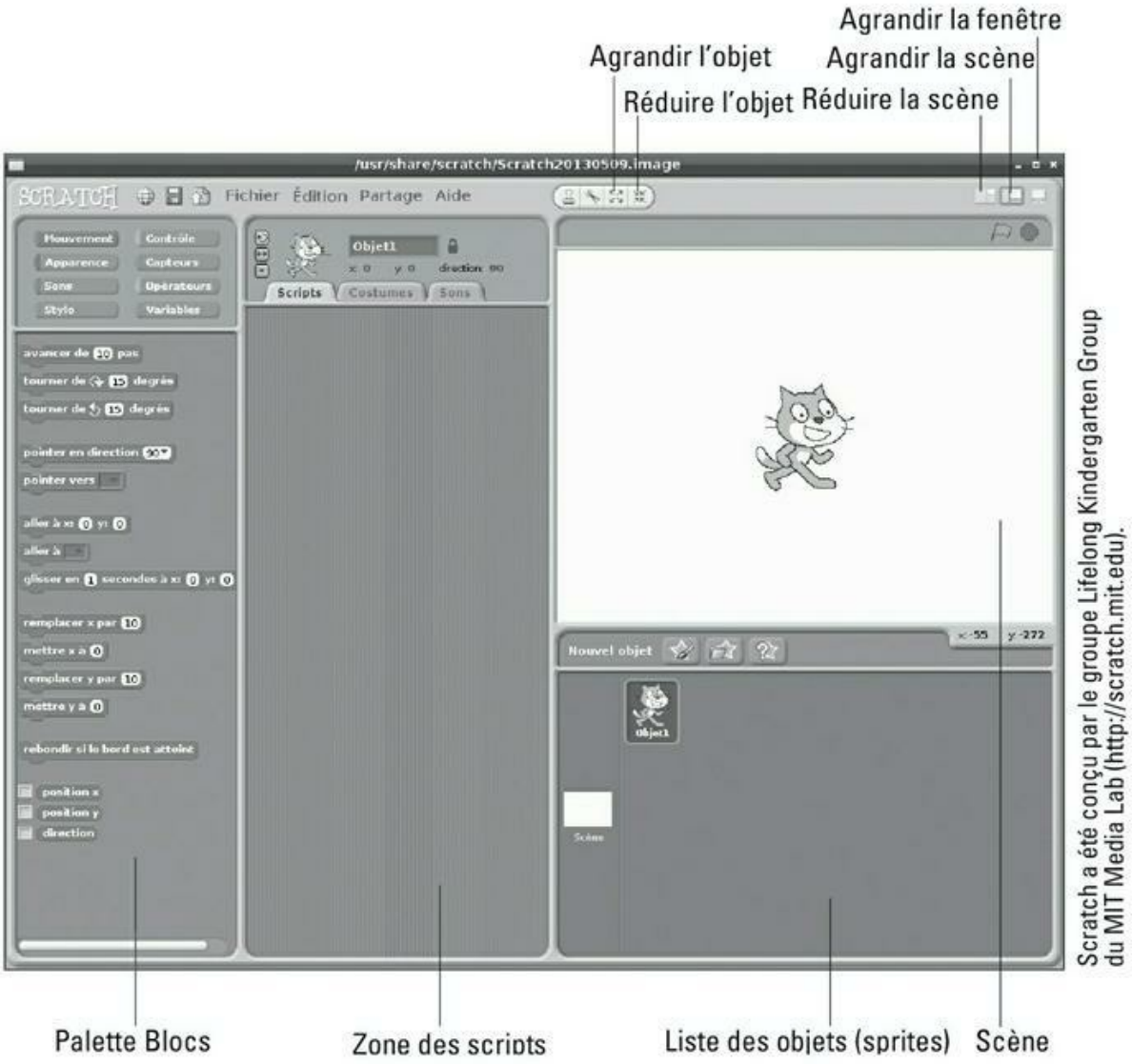


Figure 9.1 : Aspect initial de la fenêtre de l'atelier Scratch.

La **Liste des objets** est située sous la scène. Ces objets sont les éléments visuels mobiles de votre programme, par exemple le héros du jeu. Ce sont des images que vous animez en les déplaçant ou en changeant leur aspect.

Pour l'instant, il n'y a qu'un objet : un chat portant le nom **Objet1** ou **Sprite1**.

L'écriture d'un programme Scratch consiste à placer dans le volet de script, les uns sous les autres, des blocs d'instructions provenant du magasin de blocs qui est la palette **Blocs** située à gauche (ou au milieu en 2.0). Au départ, c'est la catégorie de blocs **Mouvement** qui est visible. Elle contient par exemple un bloc pour avancer de 10 pas, un autre pour tourner à droite de x degrés, *etc*.

La **Zone de scripts** est la partie centrale dans Scratch 1 (la partie de droite dans Scratch 2). C'est ici que va se construire votre programme petit à petit. L'onglet le plus utilisé est celui actuellement ouvert, **Scripts**.

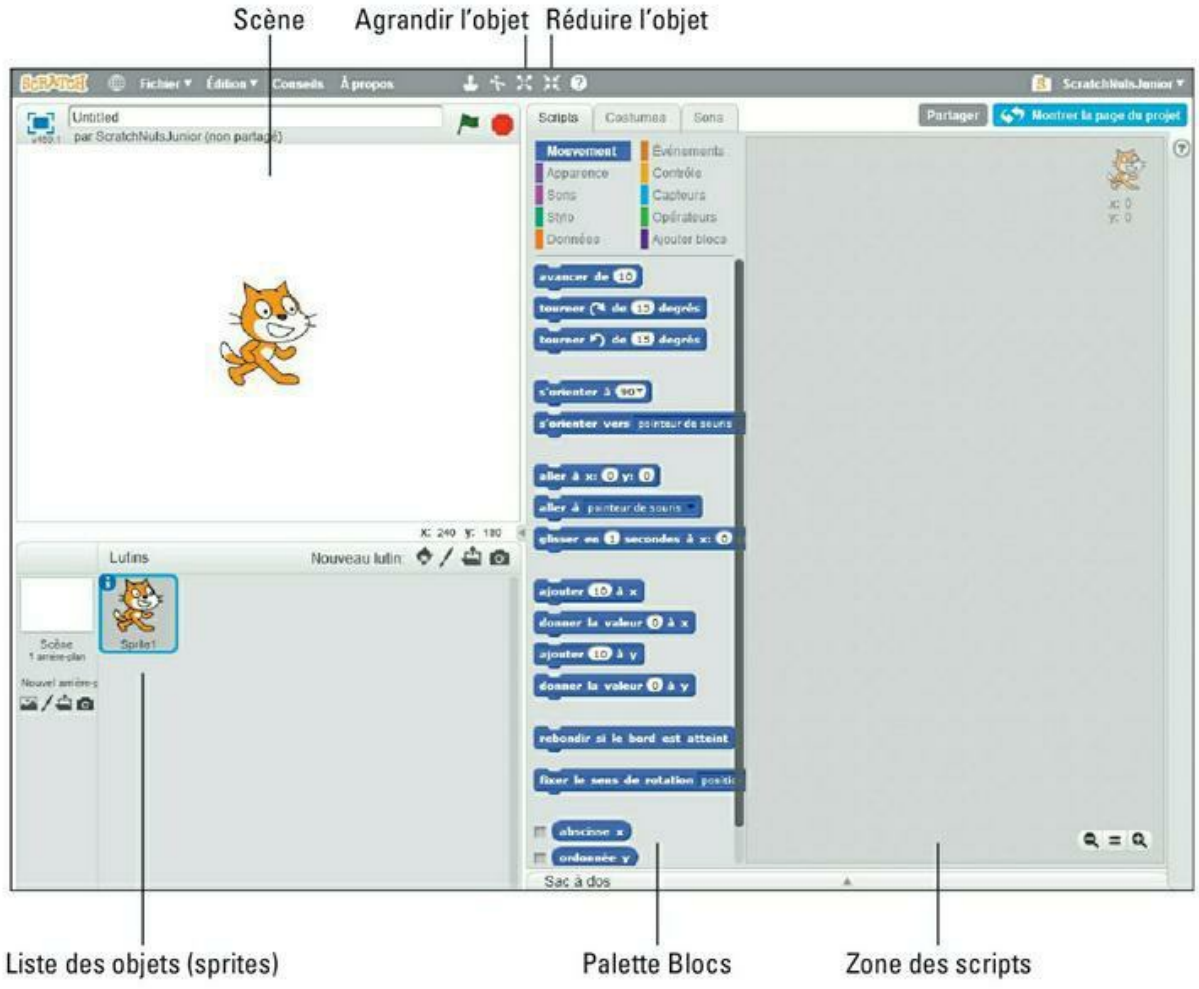


Figure 9.2 : Aspect initial de la fenêtre de l'atelier Scratch en version 2.

Les trois boutons en haut à droite (voir en Figure 9.1) permettent de jouer sur la taille de la scène (petit format, format normal et plein écran). En petit format, vous avez plus de place pour la zone de scripts, ce qui est utile quand le script devient touffu, comme nous le verrons plus loin dans ce chapitre.

Pour plus de confort, agrandissez la fenêtre de l'atelier Scratch au moyen du bouton d'agrandissement de fenêtre en haut à droite.

Positionner et redimensionner un objet

L'unique objet actuellement présent sur la scène (le chat) peut être déplacé pour le poser à l'endroit où vous voulez qu'il se trouve au début de votre programme.

Vous pouvez également le redimensionner au moyen des deux boutons au-dessus de la fenêtre de scène (voyez dans la Figure 9.1). Cliquez l'un des deux boutons. Le pointeur de souris prend l'aspect d'une quadruple flèche. Les flèches sont dirigées vers l'extérieur pour le bouton d'agrandissement et vers l'intérieur pour le bouton de réduction. Vous pouvez alors cliquer dans votre objet sur la scène plusieurs fois pour changer la taille jusqu'à aboutir à celle désirée. Quand vous en avez fini avec ces outils, cliquez n'importe où en dehors d'un objet pour ramener le pointeur à son aspect normal.

Vous pouvez prendre l'image du chat avec le pointeur de souris et la déplacer dans la scène pour placer le chat à l'endroit où vous voulez qu'il se trouve en début de programme.

Pour redimensionner l'objet, utilisez les deux boutons au-dessus de la scène (légendés en Figure 9.1). Cliquez un des boutons pour voir le pointeur changer d'aspect : flèches vers l'extérieur pour agrandir et vers l'intérieur pour réduire. Cliquez de façon répétée dans la scène pour amener l'image à la taille désirée. Pour quitter ce mode de redimensionnement, cliquez ailleurs pour redonner au pointeur son aspect normal.

Déplacer l'objet par programme

Scratch est vraiment conçu pour permettre les essais. Vous voyez la palette des blocs du côté gauche. Vérifiez que la catégorie sélectionnée est bien **Mouvement** (la première en haut). Cliquez dans le chat pour qu'il soit sélectionné, puis double-cliquez sur une des instructions, par exemple **Avancer de 10 pas**. Vous devriez voir le chat se déplacer un peu vers la droite. Vous pouvez aussi utiliser les blocs pour faire tourner l'objet de 15 degrés dans un sens ou dans l'autre.

Si votre objet va à un endroit que vous n'avez pas prévu (c'est un peu naturel chez les chats, non ?), il suffit de le sélectionner dans la scène et de le déplacer pour le ramener à l'endroit désiré. Pour corriger une rotation, cliquez sur la miniature du petit chat dans le haut de la zone centrale des scripts en maintenant enfoncé le bouton de souris puis faites tourner le pointeur en cercle. Dans Scratch 2, vous modifiez l'orientation d'un objet en cliquant le bouton **i** puis en manipulant la ligne bleue de l'indicateur de direction. Vous terminez en cliquant la flèche de retour pour réafficher la liste des objets.

Pour l'instant, vous ne pouvez pas essayer tous les blocs de commande. Certains ne fonctionnent que s'ils sont combinés à d'autres blocs ou n'ont d'intérêt que dans certaines situations. Cela ne vous empêche pas de faire des essais. Si vous cliquez quelque chose qui ne fonctionne pas, vous allez peut-être obtenir un message d'erreur, mais cela ne risque pas de faire du mal ni à Scratch, ni à votre Raspberry Pi.

Voyons donc quels genres de mouvement vous pouvez faire faire à votre objet.

Mouvements et orientation

Pour positionner et déplacer un objet, vous pouvez utiliser deux méthodes. La première consiste à faire marcher l'objet puis à changer sa direction avant qu'il marche dans cette autre direction.

Voici les cinq blocs qui servent à déplacer un objet de cette façon (voir Figure 9.3).

Figure 9.3 : Les cinq blocs de mouvement.

Avancer de 10 pas : Ce bloc fait marcher dans la direction actuelle. Si vous avez fait tourner l'objet, l'angle peut faire déplacer l'objet en diagonale sur la scène. Vous pouvez cliquer une fois dans la petite zone de saisie du bloc pour saisir une autre valeur afin d'augmenter ou de

réduire la quantité de déplacement. Sachez cependant que les grandes valeurs font perdre l'illusion d'un déplacement relatif (l'objet saute).

Tourner de 15 degrés (à droite ou à gauche) : Ce bloc fait tourner l'objet sur lui-même. Comme pour le bloc précédent, vous pouvez modifier la valeur pour contrôler le nombre de degrés de la rotation. L'objet va alors se déplacer dans la direction indiquée si vous utilisez ensuite un bloc **Avancer de 10 pas**.

Pointer en direction/s'orienter à 90 : Quelle que soit l'orientation actuelle de l'objet, ce bloc le ramène dans la direction spécifiée. Si vous ne changez pas la valeur, l'objet va être orienté vers la droite. Vous modifiez la valeur pour contrôler l'orientation de façon absolue. Sachez que les valeurs sont exprimées en degrés à partir de midi (voir Figure 9.4). La convention ressemble à celle des aiguilles d'une horloge : lorsque l'aiguille pointe sur 15 heures, cela correspond à la valeur 90 degrés. Lorsque l'aiguille indique 18 heures, cela correspond à 180 degrés. Pour la partie gauche, vous devez utiliser des valeurs négatives, par exemple -90. La petite flèche du menu associé à la zone de saisie de valeur donne accès aux quatre directions principales. Vous vous demandez peut-être s'il est possible d'indiquer la valeur 270 pour se diriger vers la gauche. Sachez que cela fonctionne, mais provoque parfois des erreurs dans les programmes. D'ailleurs, si vous orientez le chat dans la direction 270 puis demandez à Scratch quelle est son orientation, il vous répondra −90. Pour ne pas subir de telles incohérences, utilisez les valeurs 0 à 180 pour toute la moitié droite, et les valeurs – 179 à – 1 pour la moitié gauche.

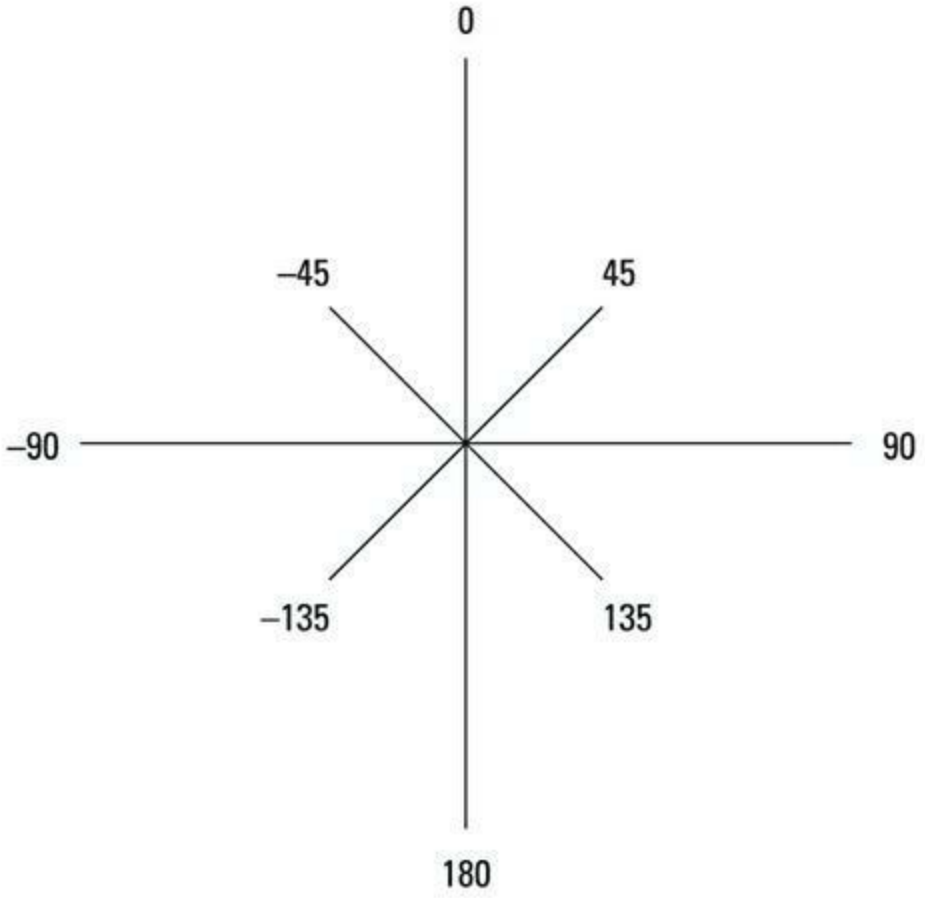


Figure 9.4 : Convention des degrés d'orientation d'un objet dans Scratch.

Pointer/s'orienter vers : Vous pouvez obliger votre objet à s'orienter en direction du pointeur de souris ou d'un autre objet. Servez-vous du petit menu de ce bloc pour choisir vers quoi l'objet doit viser.

Après avoir changé la valeur d'un bloc, pensez à double-cliquer à nouveau dans le bloc pour effectuer un test.

Les coordonnées dans la scène

Vous pouvez également déplacer votre objet dans la scène en utilisant les coordonnées, ce qui rend plus facile le placement de l'objet en un endroit exact. La position initiale de l'objet n'a dans ce cas aucune importance.

Chaque point dans la scène possède un jeu de coordonnées : une position X (dans le sens horizontal) et une position Y (dans le sens vertical). Les valeurs en X vont de – 240 pour le côté gauche à 240 pour le côté droit. Les valeurs en Y correspondent à – 180 pour le bord inférieur de la scène et à 180 pour le bord supérieur. Vous en déduisez aisément que la scène contient 480 unités en largeur sur 360 unités en hauteur. Le point

d’origine correspond au centre de la scène, et c’est là que se trouve votre chat au départ. À cet endroit, X et Y valent 0. La Figure 9.5 constitue une référence visuelle de ce système de coordonnées.

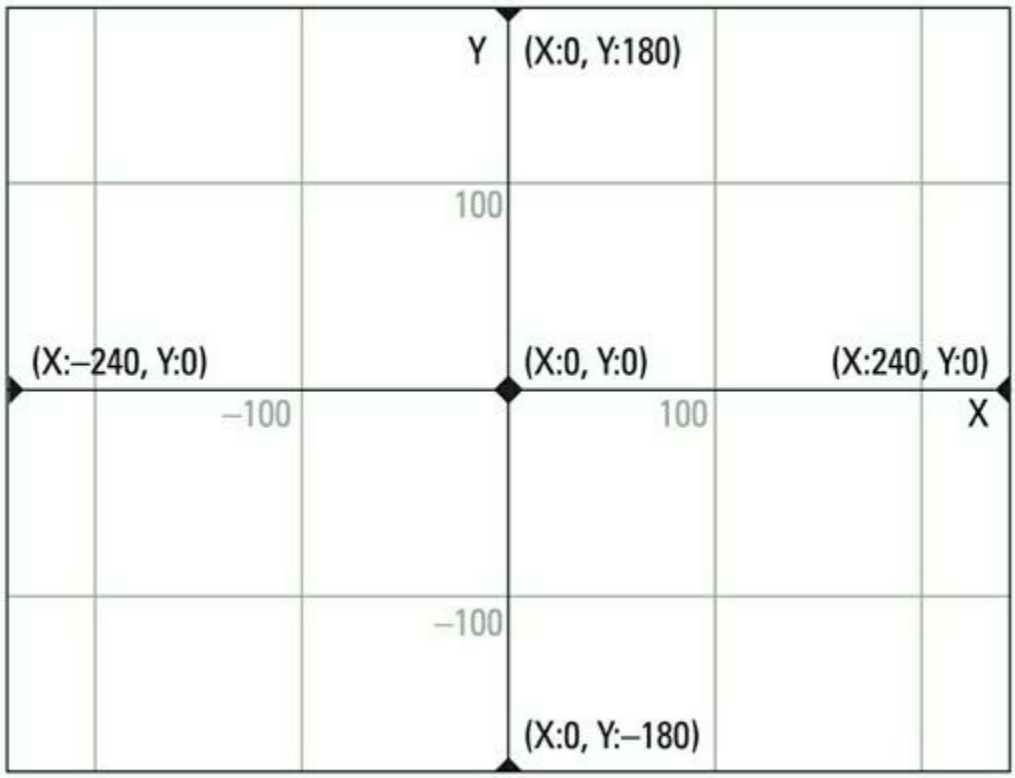


Figure 9.5 : Le système de coordonnées dans la grille de la scène.

Lorsque vous déplacez le pointeur de souris dans les limites de la scène, vous pouvez lire les coordonnées de la position actuelle à droite sous la scène.

Plusieurs blocs de la catégorie Mouvement se fondent sur les coordonnées en X et en Y (voyez la Figure 9.6) : ce sont les blocs de mouvement d’objets qui se basent sur des coordonnées dans la grille.

Aller à x : 0 y : 0 : Ce bloc vous permet d’amener directement l’objet en un point absolu de la scène. Par défaut, le point est ramené au centre (x=0, y=0). Pour choisir une autre position, vous modifiez les valeurs pour X et pour Y.

Aller à : Ce bloc permet d’amener l’objet vers une position prédéfinie telle que celle du pointeur de souris ou la position d’un autre objet.

Glisser en 1 seconde à x : 0 y : 0 : Le bloc **Aller à** fait sauter directement l’objet vers la nouvelle position. Le bloc **Glisser en** déplace l’objet en douceur, sur la durée indiquée en paramètre. Vous pouvez spécifier des valeurs fractionnaires de seconde comme par exemple **0.5** pour une demi-seconde.

Ajouter 10 à X (Remplacer X par 10) : Ce mouvement déplace l'objet de 10 unités vers la droite. Vous pouvez bien sûr changer la valeur du décalage, et même spécifier une valeur négative pour aller à gauche. Sachez que cette commande ne modifie pas la position de l'objet dans le sens vertical et qu'elle ne tient pas compte de l'orientation de l'objet.

Donner la valeur 0 à X (Mettre X à 0) : Ce bloc redéfinit la position horizontale de l'objet de façon absolue dans la scène, sans modifier sa position verticale. Avec la valeur 0, l'objet est ramené au centre dans le sens horizontal. Vous pouvez bien sûr modifier la valeur pour le déplacer vers la droite ou vers la gauche.

Une valeur négative concerne la moitié gauche de la scène et une valeur positive la moitié droite.

Ajouter 10 à Y (Remplacer Y par 10) : Ce bloc permet un déplacement relatif dans le sens vertical de l'objet, sans modifier sa position dans le sens horizontal et sans tenir compte de l'orientation de l'objet. Une valeur négative déplace l'objet vers le bas de la scène.

Donner la valeur 0 à Y (Mettre Y à 0) : Ce bloc redéfinit la position verticale absolue de l'objet dans la scène sans modifier sa position horizontale, ni tenir compte de son orientation. Une valeur positive correspond à la partie supérieure de la scène et une valeur négative à la partie inférieure.

pointer en direction 90
pointer vers
aller à x: 0 y: 0
aller à
glisser en 1 secondes à x: 0 y: 0
remplacer x par 10
mettre x à 0
remplacer y par 10
mettre y à 0

Figure 9.6 : Les blocs de repositionnement absolu dans la scène d'après les coordonnées.

Rappelons qu'il faut cliquer deux fois dans un bloc pour tester son effet sur l'objet.

Afficher les informations d'objet

Il devient parfois difficile de bien suivre la position actuelle de l'objet et son orientation. Vous pouvez faire afficher les valeurs courantes de la position en X et en Y et de l'orientation. Servez-vous à cet effet des blocs tout en bas de la palette des blocs (Figure 9.7). En les utilisant, vous ralentissez un peu le programme et cela occupe un peu d'espace à l'écran, mais ils peuvent constituer des outils indispensables pour faire des tests pendant la conception d'un jeu.

Figure 9.7 : Les blocs pour afficher des informations sur l'objet dans la scène.

Modifier l'aspect d'un objet

Vous pouvez déplacer votre objet dans la scène, mais vous pouvez également modifier son aspect.

Utiliser des costumes

Vous pouvez comparer les objets aux personnages dans un jeu vidéo (même si les objets peuvent servir à bien d'autres choses, et notamment aux obstacles). À chaque objet peuvent correspondre un certain nombre de costumes qui sont des images différentes de cet objet. Lorsque vous créez plusieurs costumes sur le même modèle, cela vous permet de créer une illusion d'animation par simple basculement de l'un vers l'autre. L'objet initial (**Objet1**, le chat) possède déjà deux costumes. Lorsque vous basculez rapidement entre les deux costumes, vous donnez l'illusion que le chat est en train de courir.

Pour voir tous les costumes de l'objet actuel, cliquez l'onglet **Costumes** dans le haut de la zone des scripts (Figure 9.8). Pour modifier l'aspect de l'objet Chat, utilisez le bouton **Édition** du costume désiré. (Dans Scratch 2, cliquez l'objet et faites vos retouches à droite.)

Vous pouvez également créer un nouvel aspect de l'objet au moyen du bouton **Dupliquer** du costume désiré (une image de tampon encreur). Vous modifiez l'aspect de cette copie du costume. (Dans Scratch 2, cliquez droit dans l'objet et choisissez Dupliquer.)

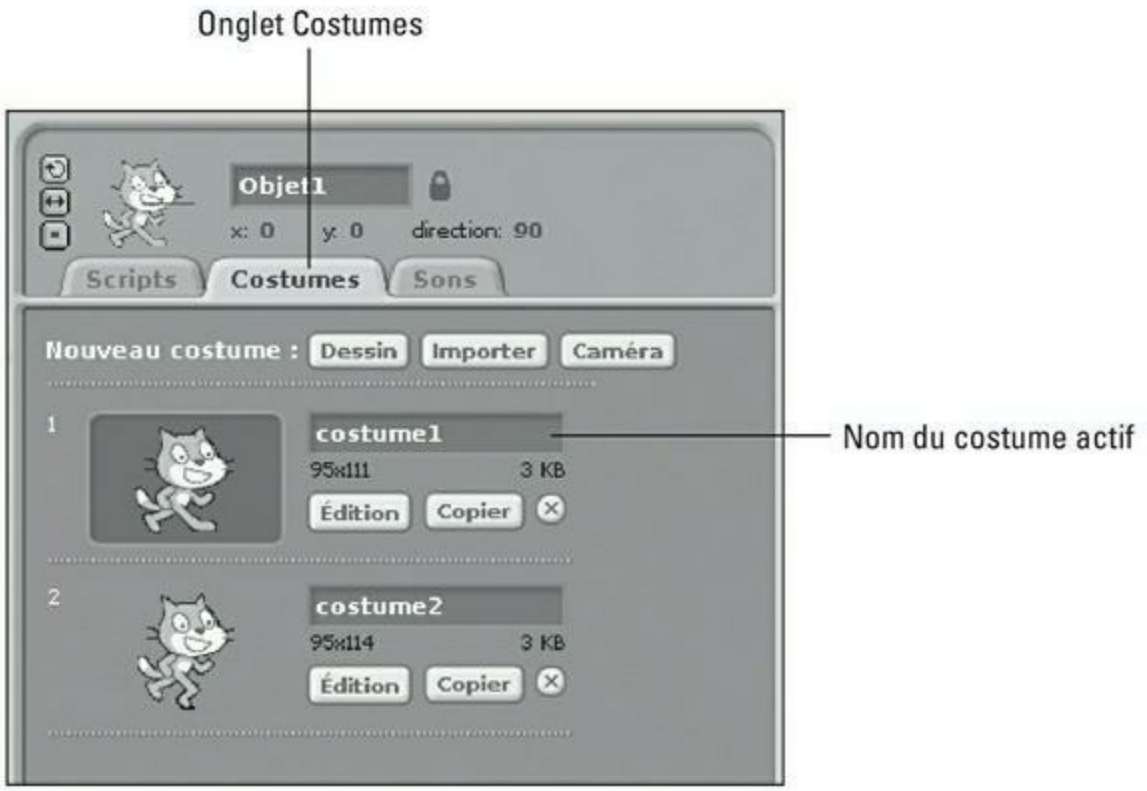


Figure 9.8 : Vous pouvez modifier l'aspect d'un objet en créant un nouveau costume.

Tant que vous ne faites que découvrir l'utilisation des objets de Scratch, cela n'a pas trop d'importance, mais il est bon de prendre l'habitude de donner des noms suggestifs à vos objets. Vous aurez beaucoup moins de mal à vous souvenir qu'un costume portant le nom **Game Over** est destiné à être affiché lorsque le joueur a perdu que si vous aviez donné le nom **Costume7**. Pour changer le nom d'un costume, cliquez sur l'onglet **Costumes** pour afficher tous les costumes puis sélectionnez le nom actuel du costume et saisissez le nouveau nom (voir Figure 9.8).

Dans la palette des blocs, la deuxième catégorie de blocs, **Apparence**, comporte deux blocs permettant de basculer d'un costume à un autre (voir Figure 9.9).

Basculer sur le costume xxx : Lorsque vous connaissez le costume que vous voulez afficher, vous choisissez directement son nom dans le menu de ce bloc, puis vous double-cliquez le bloc.

Costume suivant : Chaque utilisation de ce bloc fait changer en boucle le costume de l'objet. Une fois tous les costumes affichés, le premier

costume est affiché à nouveau.

Si nécessaire, vous pouvez afficher le numéro ou le nom du costume actuel d'un objet sur la scène, ce qui peut être utile pour les tests. Il suffit de cocher la case de l'option **Costume n°**.

Utiliser les bulles Dire et Penser

Tous les blocs de commande qui ont un effet sur l'aspect d'un objet sont réunis dans la catégorie **Apparence** de la palette des blocs (voir Figure 9.9).

Figure 9.9 : Sélection de blocs d'apparence permettant de contrôler l'aspect d'un objet.

Scratch propose quatre blocs pour faire parler ou penser l'objet en affichant un phylactère (voyez dans les Figures 9.9 et 9.10). Ces blocs permettent donc d'afficher un message pour le joueur ou pour l'utilisateur (vous pouvez bien sûr décider du contenu du message qui est au départ *Salut !* ou *Mmmh*). Dans la Figure 9.10, vous pouvez voir quatre bulles **Dire** et quatre bulles **Penser**.

Si vous indiquez une durée, l'objet va se mettre en pause et le message ne disparaîtra qu'à la fin de cette durée.

En revanche, si vous ne spécifiez pas de durée, vous ferez disparaître la bulle en utilisant le même bloc **Dire** ou **Penser**, mais en ayant pris soin d'effacer tout texte de message.

Les effets graphiques

La catégorie de blocs **Apparence** propose un certain nombre d'effets graphiques applicables à un objet. Dans la Figure 9.9, vous pouvez voir 8 effets appliqués au chat sur la scène. L'effet **Couleur** change la palette de couleurs de l'objet, en transformant par exemple les orange en verts. L'effet **Œil de poisson** ressemble aux optiques à focale courte portant ce nom (*Fisheye*) ; le résultat est un grossissement de la partie centrale de l'objet. L'effet **Tournoyer** fait tourner les pixels de l'objet autour de son centre. L'effet **Pixeliser** réduit la netteté graphique de l'objet. L'effet **Mosaïque** réduit l'objet puis le répète sous forme de motifs dans l'espace de départ. Les deux effets **Luminosité** et **Fantôme** sont très proches, sauf que le premier augmente l'intensité des couleurs (le noir du contour du chat devient argenté et l'orange est estompé). L'effet **Fantôme** estompe toutes les couleurs de façon régulière.

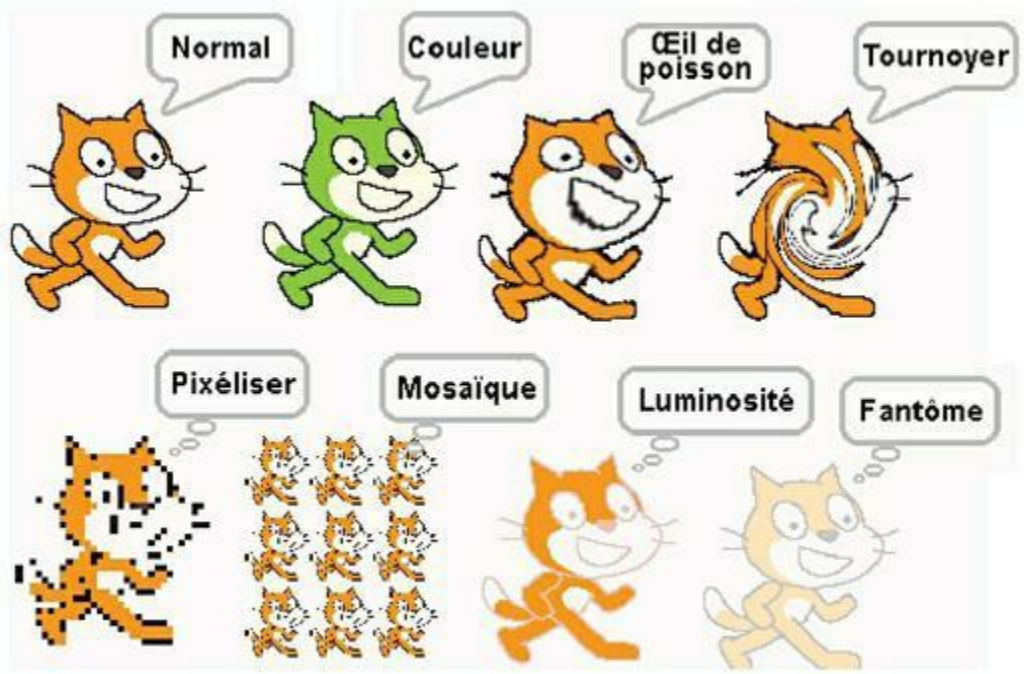


Figure 9.10 : Les différents effets graphiques applicables à un objet.

Trois blocs permettent de contrôler les effets graphiques :

Modifier l'effet xxx par 25 : Le petit menu permet de décider quel effet graphique vous voulez appliquer (par défaut, c'est l'effet **Couleur**). Vous pouvez indiquer la quantité d'effet en pourcentage (25 % par défaut). Une valeur négative réduit encore la quantité d'effet appliquée à l'objet.

Mettre l'effet xxx à 0 : Ce bloc permet de modifier la quantité d'effet appliquée. La valeur 0 désactive l'effet. Les sept effets peuvent être utilisés avec ce bloc.

Annuler les effets graphiques : Ce bloc très simple supprime tous les effets graphiques actuellement en vigueur pour un objet pour qu'il retrouve son aspect initial.

Sachez que les effets graphiques donnent de super résultats, mais qu'ils ralentissent le programme, surtout sur les anciennes cartes Raspberry comme la modèle B+. Vous les utiliserez avec modération à des moments particuliers de l'animation ou du jeu. Si vous en abusez, ils vont réduire la vivacité des réponses de l'ordinateur.

Redimensionner un objet

Un peu plus haut dans le même chapitre, nous avons montré comment modifier la taille initiale d'un objet sur la scène. Deux blocs vous permettent de contrôler cette taille, par exemple pour faire grossir l'objet au fur et à mesure que vous progressez dans un jeu.

Voici les deux blocs permettant de contrôler la taille d'un objet.

Modifier la taille par 10 : Ce bloc sert à modifier relativement la taille de l'objet d'un certain nombre d'unités. Vous pouvez bien sûr modifier la valeur associée. Une valeur négative réduit la taille au lieu de l'augmenter.

Mettre la taille à xxx % : Avec ce bloc, vous pouvez changer l'échelle d'un objet. La valeur par défaut qui est égale à 100 % supprime tous les changements d'échelle appliqués.

Comme pour les catégories précédentes, vous disposez d'une option comportant une case à cocher et intitulée **Taille**. Elle permet d'afficher la taille actuelle de l'objet sur la scène. Cette option vous sera utile lorsque vous ferez vos tests.

Contrôler la visibilité d'un objet

Dans certains cas, vous voudrez qu'un objet ne soit pas visible sur la scène, sans le supprimer. Par exemple, si votre jeu consiste à détruire des vaisseaux spatiaux, le vaisseau venant d'être détruit doit être masqué. Voici les deux blocs qui permettent de contrôler la visibilité d'un objet.

Cacher : Ce bloc rend l'objet concerné invisible sur la scène. Une fois qu'un objet est ainsi masqué, Scratch n'en tient plus compte dans la

détection de ses contacts avec d'autres objets. Vous pouvez cependant déplacer un objet caché afin qu'il réapparaisse en un autre endroit lorsque vous voudrez à nouveau le montrer.

Montrer : Les objets sont visibles par défaut, mais ce bloc vous permet de rendre à nouveau visible un objet que vous auriez caché.

Il arrive que certains objets se chevauchent les uns les autres. Vous disposez d'un bloc **Envoyer au premier plan** pour que l'objet sélectionné soit affiché devant tous les autres. Inversement, pour renvoyer un objet derrière, vous utilisez le bloc **Déplacer de X plans arrière**.

Effets sonores et musique

De même que vous pouvez contrôler l'aspect de chacun de vos objets, vous pouvez y associer des effets sonores. Scratch est livré avec plusieurs collections d'effets dans les domaines des cris d'animaux, des bruits humains, des instruments de musique, des percussions ou de quelques effets électroniques. La plupart de ces bruitages conviendront bien pour sonoriser les objets dans Scratch.

$À l'heure où nous écrivons ces lignes, certains des fichiers sont fournis au format MP3, mais Scratch ne peut lire que ceux au format WAV. Si vous voyez un message comme quoi un son est dans un format inconnu, essayez un autre son.

Voici les deux étapes permettant de sélectionner dans votre projet Scratch d'autres sons que le son minimal *Miaou* :

Vous devez d'abord importer le son pour votre objet.

1. **Dans la zone des scripts (Figure 9.11),** cliquez sur l'onglet **Sons** puis le bouton **Importer**.

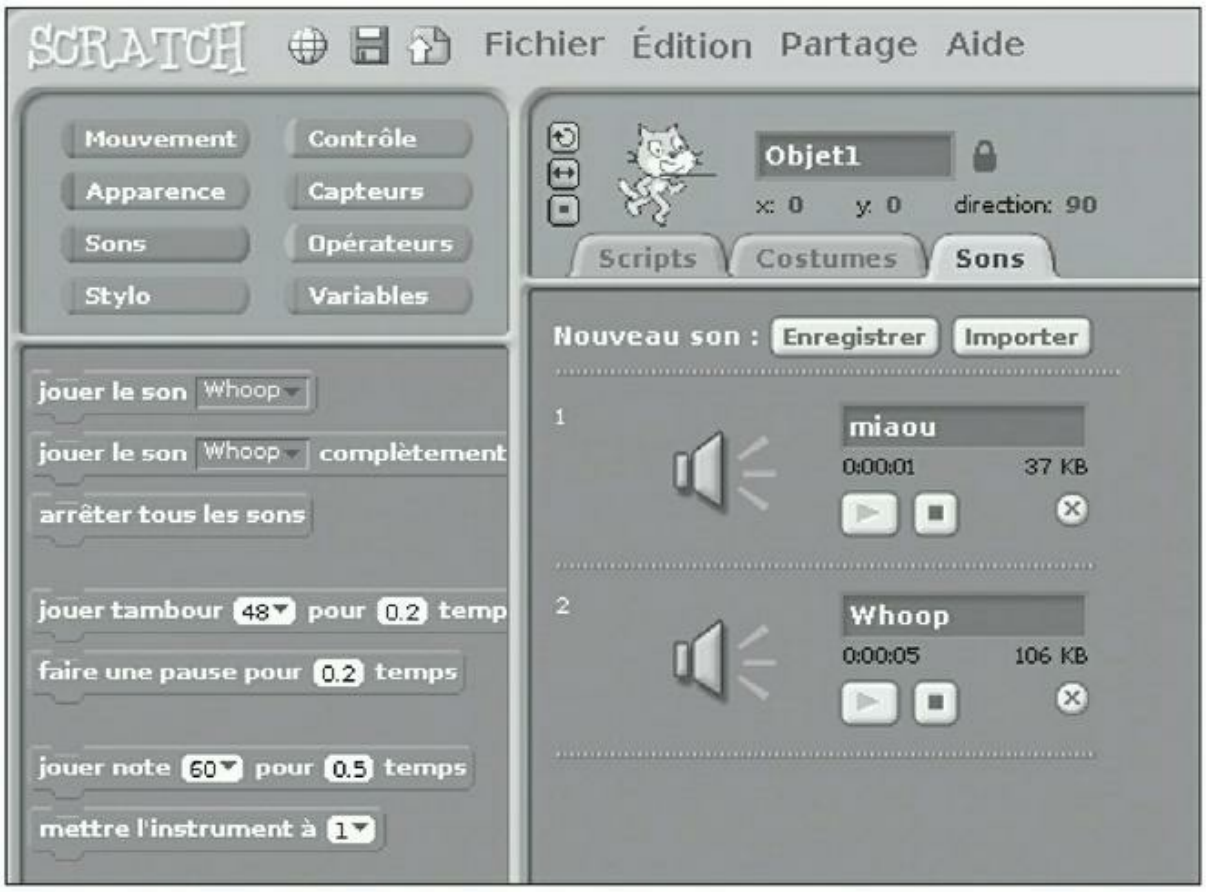


Figure 9.11 : Ajout d'un effet sonore à un objet.

2. **Naviguez parmi les catégories de sons**.
3. **(Facultatif) Vous pouvez cliquer dans un fichier pour en entendre un aperçu**. (Dans Scratch 2, utilisez le bouton de lecture à côté du nom.)
4. **Double-cliquez pour importer le son**.
5. **(Facultatif) Une fois que le son est importé, il suffit d'utiliser le bouton de haut-parleur pour le préécouter**.

Le bouton **X** permet de supprimer le son du projet. Sachez que le fichier du son reste inchangé sur la carte mémoire SD, ce qui permet de le réimporter.

Vous pouvez ensuite faire jouer le son au moyen d'un des blocs appropriés. Dans la zone des catégories de la palette des blocs à gauche, cliquez sur la catégorie **Sons**.

Le bloc intitulé **Jouer le son xxx** comporte un menu dans lequel vous pouvez choisir parmi les sons que vous avez importés. Le bloc **Jouer le son complètement** provoque une pause de tous les déplacements de l'objet et de toutes les actions des autres blocs pour cet objet jusqu'à ce que la lecture du son soit terminée.

Nous verrons dans le Chapitre 10 comment utiliser plusieurs objets dans un même projet. Les effets sonores sont toujours importés par rapport à un objet. Si vous ne voyez pas dans le petit menu le nom du son, vérifiez que vous avez bien importé le son lorsque l'objet désiré était sélectionné.

Vous disposez d'autres blocs sonores qui permettent de créer de la musique avec Scratch, en utilisant des percussions et des instruments. Les notes sont numérotées. Le do correspond à la valeur 60, le do dièse à 61, le ré à 62, *etc*. Le bloc nommé **Jouer note 60 pour 0.5 temps** sert à émettre une note pendant un certain temps. Lorsque vous ouvrez le menu de ce bloc, vous voyez apparaître un piano qui permet de choisir la note à jouer.

Si vous ne connaissez rien en musique, cantonnez-vous au do, et ne sélectionnez que les touches blanches. Évitez également de faire des sauts trop importants sur le clavier.

Vous disposez d'une variante de ce bloc nommée **Mettre l'instrument à 1** qui permet de changer d'instrument, mais pour en profiter dans le Scratch original, il faut d'abord installer les fichiers son correspondants. Dans la barre des tâches, cliquez le **Terminal** pour saisir dans la fenêtre les deux commandes suivantes (elles ont été présentées dans le Chapitre 5) :

```
sudo apt-get update
/usr/share/scratch/timidityinstall.sh
```

Une fois cela fait, sauvegardez vos travaux en cours et refermez les applications, car vous devez redémarrer votre Raspberry Pi. Dans le menu général, choisissez **Shutdown** puis **Reboot**.

Les numéros des notes de Scratch sont les valeurs standard du système MIDI, les mêmes que celles utilisées par Sonic Pi et indiquées dans le Tableau 14.1 du Chapitre 14.

Créer un script

Pour l'instant, nous n'avons fait qu'activer les blocs un par un dans la palette des blocs pour exécuter l'action correspondante, mais ce n'est pas

vraiment de la programmation. En effet, vous devez cliquer sur chacun des blocs à chaque fois que vous voulez exécuter. C'est vous qui réalisez le difficile travail de se souvenir de la séquence d'instructions. De plus, l'ordinateur est ralenti par la vitesse à laquelle vous arrivez à cliquer sur les blocs.

Un programme est une série d'instructions réutilisables, que vous pouvez faire exécuter (*run*) autant de fois que désiré. Pour commencer la création d'un script, il suffit de faire glisser les blocs depuis la palette des blocs vers la zone des scripts en milieu de fenêtre. La plupart des blocs vus jusqu'ici sont dotés d'un tenon sur leur bord supérieur et d'une mortaise sur le bord inférieur, ce qui leur permet de s'imbriquer facilement, comme dans un puzzle. Ne cherchez pas à les approcher avec précision : Scratch va les rabouter automatiquement dès qu'ils seront assez proches l'un de l'autre. Vous déposez ainsi les blocs dans l'ordre approprié pour que Scratch puisse les exécuter. Vous commencez en haut de la zone et vous empilez en descendant. Cela ressemble un peu à la création d'une liste de courses ou d'une recette à faire réaliser par la machine.

Une fois que vous avez créé un groupe de blocs dans la zone des scripts, vous obtenez un script dont vous pouvez lancer l'exécution en cliquant n'importe où dans le groupe. Vous voyez à ce moment la bordure clignoter en blanc et vous devriez voir l'objet (le chat) se déplacer sur la scène selon vos instructions.

Vous pouvez créer plusieurs scripts dans la zone de scripts. Cela permet par exemple de créer un premier script pour faire marcher le chat vers la gauche et un autre pour le faire marcher vers la droite, par exemple. Lorsque vous allez ajouter plusieurs objets sur la scène (nous verrons cela dans le Chapitre 10), il y aura une zone de script pour chaque objet, contenant un ou plusieurs scripts qui ne s'appliqueront qu'à cet objet.

Lorsque vous avez besoin de réorganiser le contenu de la zone des scripts, sachez que vous pouvez déplacer tout le bloc d'un script en l'agrippant par son premier bloc en haut. Si vous déplacez un bloc vers le bas de la zone, vous le détachez des blocs qui le précèdent, mais vous déplacez en même temps tous les blocs suivants. Pour supprimer un ou plusieurs blocs, il suffit de le faire glisser à nouveau dans la palette des blocs à gauche.

Apprendre le moonwalk

Le *moonwalk* est la fameuse danse de Michael Jackson dans laquelle il semble marcher normalement, tout en se déplaçant en arrière. Vous pouvez étudier le contenu de la Figure 9.12 qui présente un script pour

faire exécuter le moonwalk par notre chat dans la scène. Les deux premières lignes ramènent le chat à sa position de départ, en plein milieu et regardant à droite. Le chat nous dit qu'il adore le moonwalk puis il exécute un petit pas glissé arrière comme Michael Jackson, en émettant le même genre de cri de joie. La commande de changement de costume permet de faire changer la position des jambes. Nous réalisons ensuite un glissement vers la gauche de 150 unités. Nous effaçons alors la bulle de paroles au moyen d'un bloc **Dire** vide. Nous finissons en rebasculant sur le premier costume qui correspond au même chat, mais avec les jambes dans la position de départ. Essayez ce petit exemple !

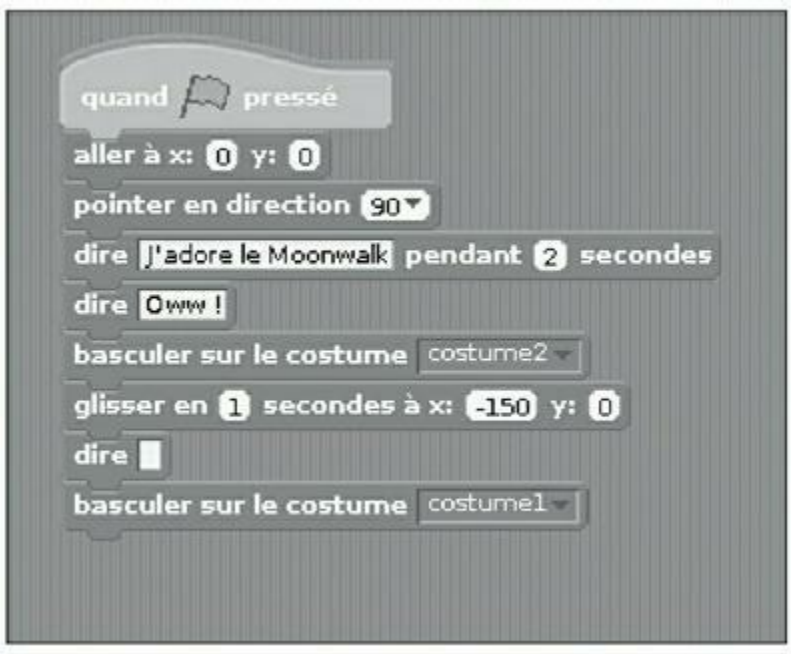

quand pressé
aller à x: 0 y: 0
pointer en direction 90
dire J'adore le Moonwalk pendant 2 secondes
dire Oww !
basculer sur le costume costume2
glisser en 1 secondes à x: -150 y: 0
dire
basculer sur le costume costume1

Figure 9.12 : Le script pour obtenir un chat exécutant le moonwalk. Ouh !

Ralentir un mouvement avec le bloc Attendre

Lorsque vous rédigez vos scripts, vous constaterez parfois que les mouvements sont trop rapides pour que l'on puisse bien les voir. Dans la palette de blocs à gauche, vous avez une catégorie intitulée **Contrôle** qui réunit des blocs jaunes. Ces blocs permettent de contrôler le déclenchement de certaines actions dans certaines conditions. Nous en parlerons plus en détail dans le prochain chapitre. Pour l'instant, intéressez-vous au bloc qui permet de créer une pause pendant un certain nombre de secondes. Ce bloc est intitulé **Attendre x secondes**. Vous pouvez le déposer dans la zone de script pour ajouter un délai afin de mieux voir l'effet de chacun des blocs. Par défaut, le délai est d'une

seconde, mais vous pouvez bien sûr spécifier n'importe quelle valeur, y compris une fraction de seconde, comme **0.5** pour une demi-seconde.

Le bloc **Dire Salut ! pendant 2 secondes** permet lui aussi d'obliger l'objet à marquer une pause avant de passer à l'exécution du bloc suivant.

Sauvegarder son travail

N'oubliez pas d'enregistrer votre travail pour pouvoir y revenir plus tard. L'option habituelle se trouve dans le menu Fichier en haut de la fenêtre, mais vous disposez également d'une icône de disquette tout en haut à gauche.

Lorsque vous êtes face à la boîte *Enregistrer* (Figure 9.13), vous remarquez plusieurs boutons du côté gauche qui permettent de choisir l'emplacement d'enregistrement du fichier. Sachez que vous n'êtes peut-être pas autorisé à écrire dans tous ces emplacements (les droits d'accès ont été présentés dans le Chapitre 5). Je conseille de rester cantonné au dossier nommé *Scratch* de votre répertoire principal Pi.

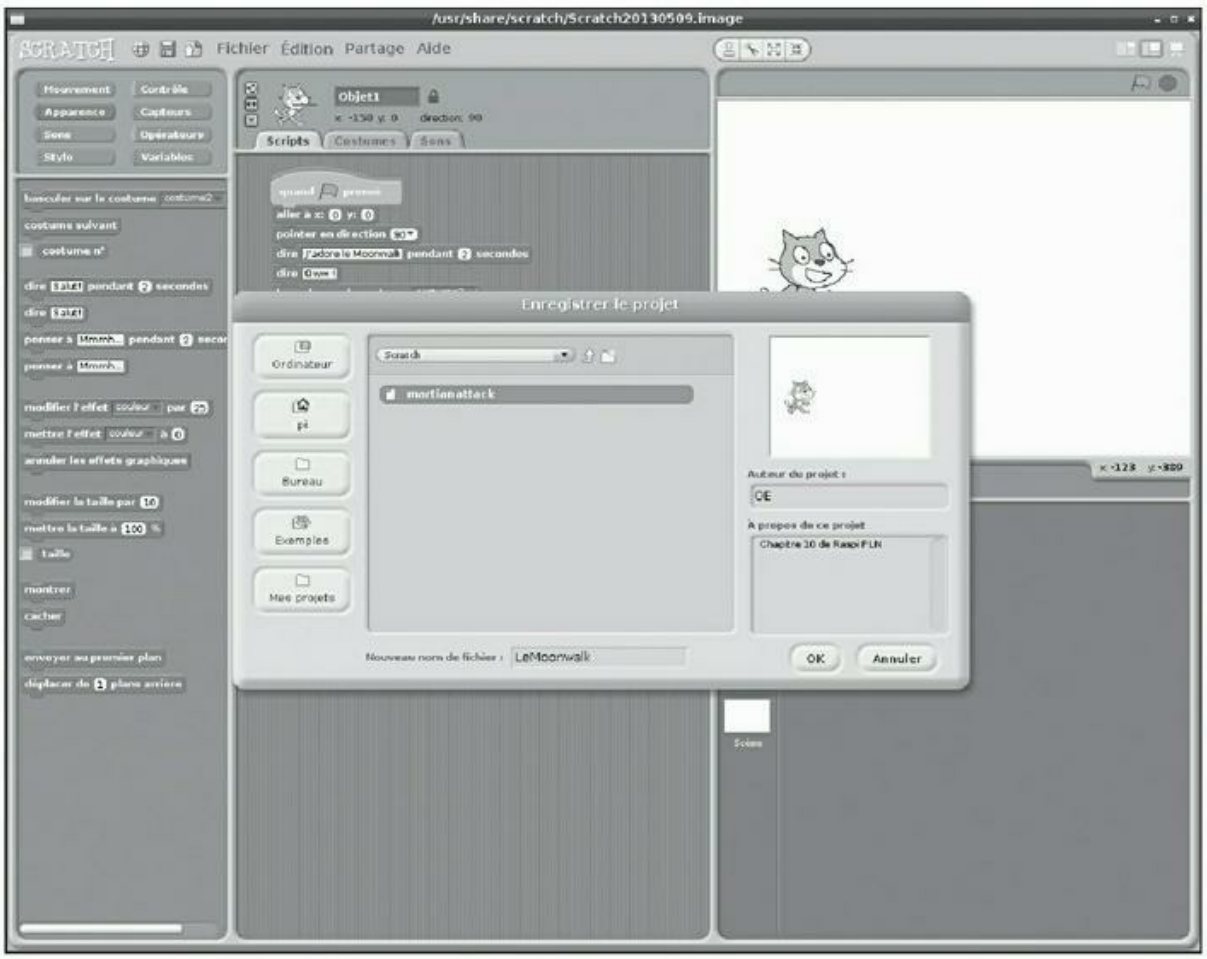


Figure 9.13 : La boîte d'enregistrement d'un script Scratch.

Dans cette boîte, vous disposez en bas à droite d'une zone dans laquelle vous pouvez indiquer votre nom et quelques notes expliquant le but de ce projet. Vous pourrez visualiser et modifier ces notes en ouvrant le menu **Fichier** pendant que vous serez en train de travailler sur ce script.

Quand vous enregistrez votre projet dans Scratch 2, la fenêtre qui apparaît permet de choisir librement où placer le fichier. Nous conseillons de ne pas vous éparpiller : stockez tout dans un sous-dossier **Scratch** de votre dossier personnel **pi**.

Nouveautés de Scratch 2

Les deux variantes de Scratch sont très similaires. Voici les quelques différences de Scratch 2 :

- **Catégorie de blocs Événement :** C'est une nouvelle catégorie qui réunit certains blocs de la catégorie **Contrôle**, par exemple ceux dont le texte commence par « Quand… » ou par « Envoyer à tous ». (La seule exception est **Quand je commence comme un clone**, qui reste dans la catégorie **Contrôle**.)
- **Clonage :** Scratch 2 permet à un lutin/objet de se dupliquer pour produire un clone relié à des instructions qui sont exécutées quand le clone est créé. Les blocs de ce groupe de clonage sont dans la catégorie **Contrôle**.
- **Catégorie Ajouter blocs :** Scratch 2 permet de créer de nouveaux blocs en combinant des blocs existants. Cela permet de rendre les scripts plus lisibles.
- **Catégorie de blocs Données :** La catégorie **Variables** a été renommée **Données** dans Scratch 2 sans aucun changement de contenu. Les variables sont présentées dans le prochain chapitre.

Malgré ces différences, les programmes écrits pour le Scratch original doivent tous fonctionner dans Scratch 2. Les guides et tutoriels restent donc utilisables. Nous donnons quelques conseils au sujet de Scratch 2 à la fin du prochain chapitre.

Chapitre 10

Programmer un jeu d'arcade en Scratch

DANS CE CHAPITRE :

- **Ajouter des effets visuels**
- **Dessiner et nommer des objets visuels**
- **Contrôler le lancement des scripts**
- **Utiliser des valeurs aléatoires**
- **Détecter la collision entre deux objets**
- **Découvrir les variables**
- **Faire se déplacer un objet automatiquement**
- **Ajouter un script à la scène**

Nous allons voir au long de ce chapitre comment se servir de Scratch pour créer un jeu d'arcade puis pouvoir y jouer. Vous pourrez personnaliser le jeu en créant vos propres objets graphiques, mais surtout, vous allez apprendre comment construire un projet de jeu, ce qui vous permettra ensuite d'en inventer d'autres.

Le principe de ce jeu consiste à faire contrôler par le joueur une soucoupe volante qui doit défendre sa planète des envahisseurs. Des aliens menaçants surgissent d'en haut et vous devez les éliminer en leur envoyant des boules de feu. Les aliens descendent progressivement, et s'ils vous touchent, vous avez perdu. Non seulement vous, mais votre planète est condamnée.

Dans ce chapitre, nous allons beaucoup utiliser la catégorie de blocs intitulée **Contrôle**, car ils vont nous permettre de coordonner les actions entre différents objets et avec le joueur. Nous supposons que vous avez une connaissance minimale de l'interface de Scratch et que vous savez

comment mettre en place des blocs pour construire un script. Si nécessaire, revoyez le Chapitre 9.

Nous allons utiliser la version originale de Scratch pour réaliser le projet de ce chapitre. Le code source est accepté dans Scratch 2, mais le jeu est très ralenti. De plus, les anciennes cartes Raspberry Pi ne permettent pas d'utiliser Scratch 2. Si vous voulez vraiment construire le projet en Scratch 2, voyez nos remarques à la fin du chapitre précédent et trouvez par vous-même où se trouvent les blocs qui ont changé de catégorie entre les deux versions.

N'hésitez pas à télécharger le fichier du programme Scratch de ce jeu sur le site de l'éditeur. (Nous avons indiqué dans l'introduction de ce livre comment accéder au fichier archive.) Le fichier qui nous intéresse se trouve dans le sous-dossier nommé **Scratch**, sous le nom **martianattack.sb**. En chargeant ce fichier, vous pourrez suivre nos explications ou vous resynchroniser si vous avez tenté de rédiger le programme pas à pas et êtes perdu. Vous vous servirez du menu **Fichier** de la fenêtre Scratch pour charger le fichier. Vous pouvez également double-cliquer dans son nom.

Lancement d'un nouveau projet Scratch

Si vous avez déjà essayé Scratch, il est possible que l'écran soit rempli de blocs et de scripts dont nous n'avons pas besoin. Démarrez un nouveau projet en ouvrant le menu **Fichier** puis en choisissant **Nouveau**.

Tout nouveau projet propose dès le départ un objet visuel qui est celui du chat que nous connaissons bien maintenant. Pour notre projet de jeu vidéo, nous n'avons pas prévu de chat. Il faut donc le supprimer (cette précaution ne s'applique pas si vous avez chargé le fichier déjà achevé). Trois techniques sont possibles :

- Sur la scène en haut à droite, cliquez droit dans le chat puis choisissez **Supprimer** dans le menu local.
- Dans la liste des objets en bas à droite, cliquez droit sur l'objet **Objet1** puis choisissez la même commande **Supprimer** dans le menu local (Figure 10.1).

» Au-dessus de la scène, cliquez dans l'icône représentant des ciseaux puis cliquez sur l'objet à supprimer dans la scène ou dans la liste des objets.

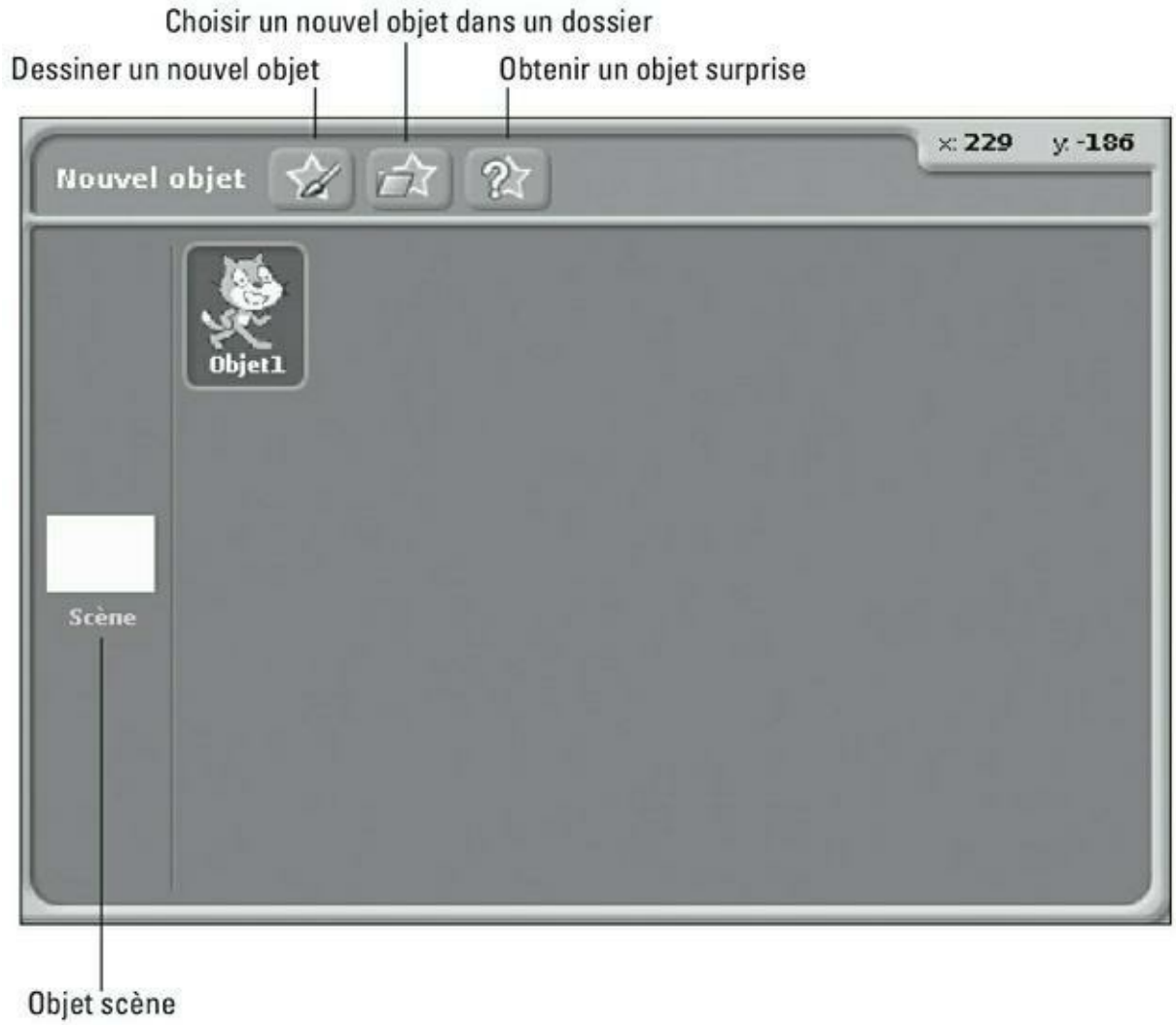


Figure 10.1 : La liste des objets avec le menu local ouvert par clic-droit sur l'objet Chat.

Soyez vigilant quand vous utilisez les ciseaux. Dans la plupart des logiciels, ils correspondent à l'action **Couper** qui permet ensuite de coller ce qui a été coupé. Dans Scratch, cela provoque la suppression de l'objet. Si vous supprimez un objet par mégarde, ouvrez le menu **Édition** et choisissez la commande **Ne pas effacer** (qui porte habituellement le nom **Annuler**).

Il ne faut pas confondre l'action de suppression d'un objet et le fait de masquer l'objet. En masquant un objet, vous ne l'enlevez pas du projet, il n'est simplement plus visible et vous pouvez le rendre à nouveau visible

au moyen d'une autre commande. En revanche, lorsque vous supprimez un objet, vous supprimez également de votre projet tous les scripts de l'objet, et tous les costumes et sons associés.

Modifier l'arrière-plan

Dans le chapitre précédent, nous avions laissé le fond de scène vide, mais vous pouvez mettre en place un fond plus intéressant. La liste des objets en bas propose une entrée nommée **Scène** (voir Figure 10.1). Cet objet spécial peut être associé à des scripts et à des images, comme les autres objets visuels. Pour la scène, les images sont des arrière-plans et non des costumes. Cliquez dans l'icône **Scène** dans la liste des objets puis cliquez dans l'onglet **Arrière-plans** tout en haut de la zone de script.

Vous pouvez dessiner un nouveau fond avec l'outil d'édition graphique intégré (nous le présentons dans la section montrant comment créer des objets dans Scratch un peu plus loin). Vous pouvez aussi importer un fichier image. Scratch est livré avec un certain nombre de fonds, mais vous pouvez aussi choisir une de vos photos. Scratch est capable d'ouvrir des images dans les trois formats .jpg, .gif ou .png.

Pour notre exemple, nous sommes partis d'une photo prise par l'auteur sur l'île de Lanzarote. Ce paysage lunaire conviendra très bien à l'ambiance de notre jeu galactique.

Ajouter des objets visuels

Un langage de programmation qui ne permettrait de travailler qu'avec des images de chat ne serait pas très intéressant pour créer des jeux vidéo. (Cela dit, lorsque l'on voit la popularité des vidéos de chats sur le Web, c'est encore discutable.) En tout cas, Scratch vous offre trois techniques pour incorporer de nouveaux objets visuels dans votre programme. Les boutons correspondants se trouvent dans le haut de la liste des objets, comme le montre la Figure 10.1.

- **Dessiner un nouvel objet :** Cette commande démarre l'éditeur graphique intégré, ce qui vous permet de créer votre objet directement dans Scratch.
- **Choisir un nouvel objet dans un dossier :** Ce bouton permet de sélectionner un des objets

prédéfinis ou d'importer un graphique que vous aurez créé avec un autre logiciel graphique. Scratch est déjà livré avec toute une série d'objets, y compris des danseurs, des hippopotames volants et des dragons crachant des flammes (que j'apprécie particulièrement !).

» **Obtenir un objet surprise :** En manque d'inspiration ? Avec ce bouton, votre créativité va être sollicitée, car il sélectionne au hasard un des objets prédéfinis livrés avec Scratch. C'est une manière très efficace de démarrer lorsque vous avez besoin de faire des tests de scripts. Si l'objet qui apparaît ne vous convient pas, il suffit de le supprimer et d'en demander un autre.

Dessiner un objet visuel dans Scratch

Une solution évidente pour donner une touche personnelle à votre jeu vidéo consiste à dessiner les objets visuels. Même si le principe du jeu est très connu, l'aspect sera vraiment original si vous créez vos images. La Figure 10.2 montre l'éditeur graphique de Scratch en pleine action.

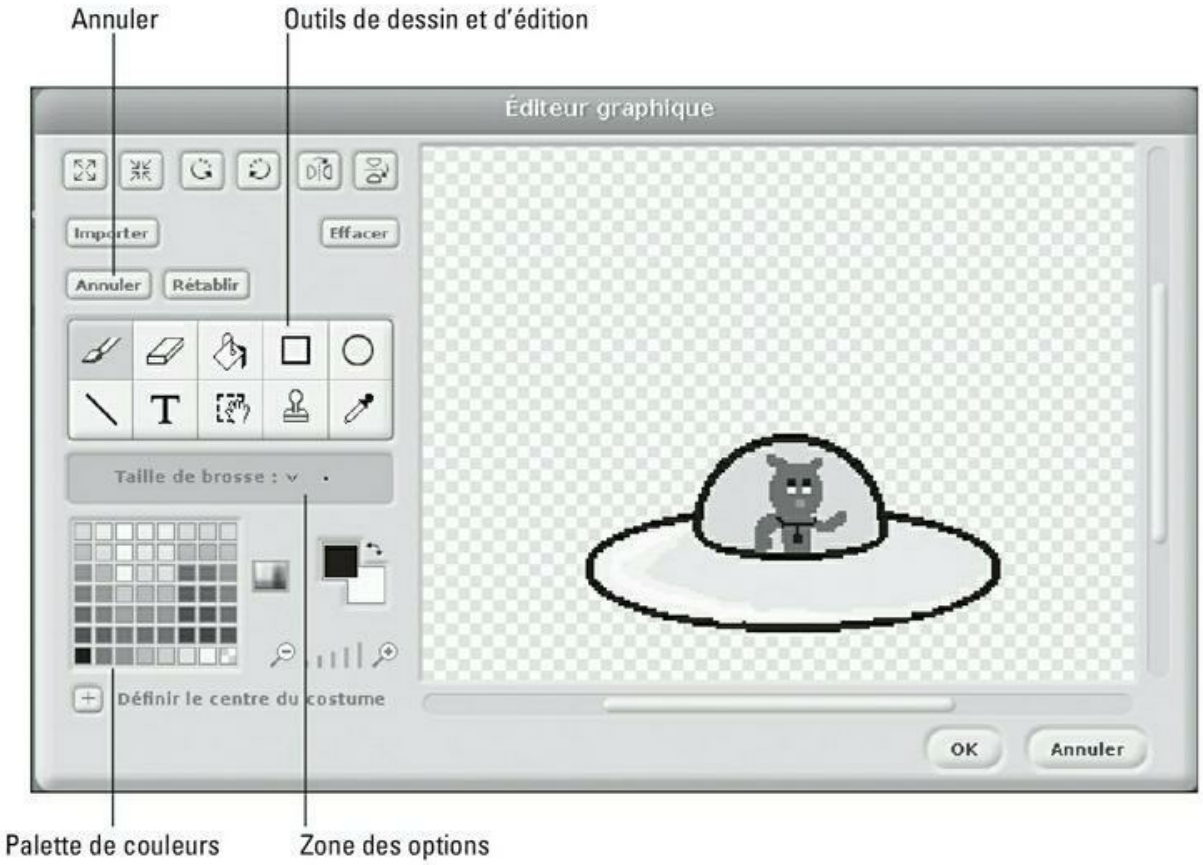


Figure 10.2 : L'éditeur graphique de Scratch.

La grande surface à droite est le canevas. Le motif à damier du fond signifie que la couleur à cet endroit est transparente. Normalement, tout ce qui se trouve en dehors de votre sujet devrait rester transparent, et seul le contenu ne doit pas l'être. Dans le coin inférieur gauche de la fenêtre (Figure 10.2), choisissez vos couleurs dans la palette (la couleur de transparence correspond au godet inférieur droit de cette palette).

Au-dessus de cette palette se trouve la barre des outils de dessin et d'édition. Vous activez un outil en cliquant dans son icône. L'icône de l'outil actif est affichée en bleu pour vous rappeler quel outil vous utilisez. Juste sous les outils se trouve la zone des options qui varient selon l'outil sélectionné. Passons en revue les différents outils de gauche à droite et de haut en bas.

- **Pinceau :** Pour tracer, vous amenez le pointeur dans le canevas et vous maintenez enfoncé le bouton principal de souris. La zone des options permet de choisir la taille de la brosse.
- **Gomme :** Vous cliquez pour effacer un endroit de l'image et vous maintenez le bouton enfoncé

tout en déplaçant pour effacer toute une zone. Pour un travail de précision, vous ne devez pas trembler. La zone des options permet de choisir la taille de la gomme.

- **Pot de peinture :** Vous cliquez avec cet outil à l'intérieur d'une forme fermée de l'image pour la remplir avec la couleur active. La zone des options permet de choisir un motif dégradé. Vous cliquez droit dans la palette des couleurs pour choisir une deuxième couleur pour le dégradé.
- **Rectangle :** Cliquez et maintenez le bouton pour marquer le premier coin du rectangle puis faites glisser avant de relâcher. La zone des options permet de choisir entre un rectangle rempli ou vide.
- **Ellipse :** L'utilisation est la même que pour l'outil rectangle. Vous cliquez pour désigner une première tangente puis vous faites glisser avant de relâcher. Les options permettent de remplir la forme ou pas. Notez que vous pouvez faire un arc de cercle parfait en traçant une ellipse puis en en effaçant une partie.
- **Ligne :** Cliquez et maintenez pour commencer à tracer, déplacez et relâchez. Les options permettent de choisir l'épaisseur du trait.
- **Texte :** Sachez que vous ne pouvez pas choisir où placer le texte dans le canevas, mais que vous pouvez forcer le passage à la ligne avec la touche Entrée. Les options permettent de choisir parmi les polices et tailles de caractères.

- **Sélection :** Cet outil permet de sélectionner une zone rectangulaire que vous voulez modifier, déplacer ou supprimer. Comme pour un rectangle, cliquez et maintenez puis glissez et relâchez. Vous pouvez ensuite déplacer la zone ailleurs dans l'image ou utiliser les boutons dans le haut de l'éditeur pour agrandir ou réduire, faire tourner dans les deux sens, ou basculer en miroir toute la région. Vous pouvez même supprimer la sélection avec la touche Suppr de votre clavier.
- **Tampon :** Cet outil permet de copier puis coller une portion de l'image. Cliquez et maintenez puis glissez pour délimiter la zone rectangulaire. Une copie de la zone choisie suit ensuite le pointeur. Vous cliquez pour tamponner en collant à la position désirée dans le canevas.
- **Pipette :** Cet outil permet de choisir comme couleur courante une des couleurs déjà présentes dans le canevas. Il vous sera utile lorsque vous aurez besoin de retrouver la même teinte, car ce sera plus rapide que de devoir la retrouver manuellement.

Le bouton **Effacer** vide totalement le canevas sauf le texte, quelle que soit la sélection actuelle. En cas d'erreur, utilisez le bouton **Annuler** (Figure 10.2).

Une fois que votre dessin vous convient, pensez à cliquer en bas à gauche dans le bouton **Définir le centre du costume** puis cliquez à l'endroit dans le canevas que vous considérez comme le centre de votre objet. C'est en effet ce point qui servira de point de rotation lorsque vous ferez tourner l'objet dans votre jeu.

Lorsque le résultat vous convient, refermez l'éditeur graphique au moyen du bouton **OK**.

Pensez à sauvegarder fréquemment votre jeu. Nous conseillons de faire une copie portant un autre nom de fichier pour chaque étape importante de votre développement. Cela vous permettra de revenir en arrière au cas où vous tomberiez sur une erreur inattendue. Cela vous évitera également de perdre un trop grand volume de travail au cas où le fichier deviendrait illisible (cela m'est arrivé une fois lorsque j'ai créé ce jeu !).

Vous pouvez à tout moment modifier votre objet en cliquant dans l'onglet **Costumes** de votre objet (voir Figure 10.3) puis en utilisant le bouton **Édition** du costume désiré. Pour créer d'autres costumes pour le même objet, vous utilisez le bouton **Dessin** de l'onglet **Costumes**.

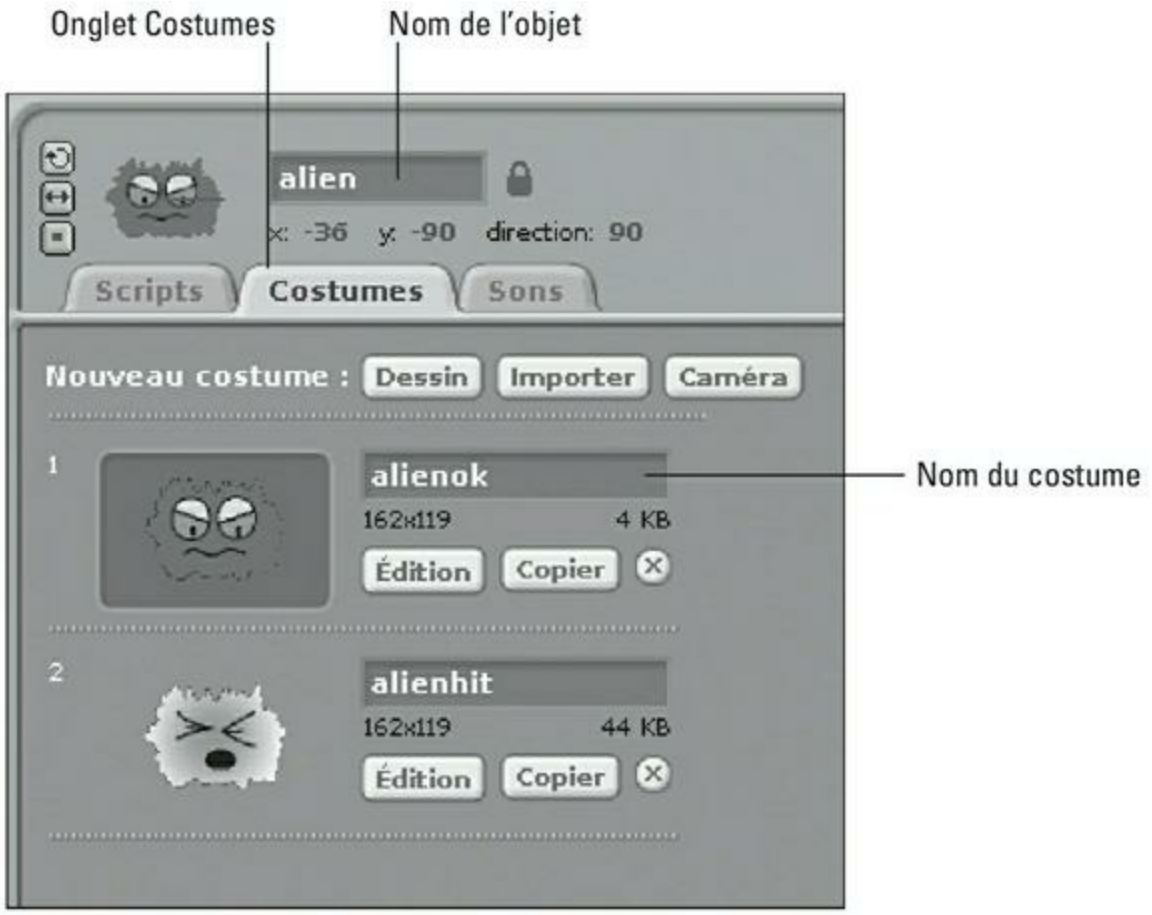


Figure 10.3 : L'onglet des costumes d'un objet visuel.

Nommer les objets

Lorsque vous commencez à programmer, vous avez intérêt à prendre au plus tôt l'habitude d'attribuer des noms intelligents à vos objets pour que vous et les autres programmeurs compreniez plus facilement ce que font vos programmes. Au départ, Scratch attribue à vos objets visuels des noms du type **Objet1** et **Objet2**, mais vous avez tout intérêt à les personnaliser. Pour renommer un objet, vous cliquez dans son nom au-

dessus de la zone de script (Figure 10.3) et saisissez le nouveau nom. Les costumes portent au départ les noms **Costume1**, **Costume2**, *etc*. Lorsque vous utilisez plusieurs costumes pour un objet, pensez à personnaliser leurs noms pour mieux les reconnaître. Pour ce faire, accédez à l'onglet des costumes et cliquez dans le nom du costume désiré puis saisissez le nouveau nom (Figure 10.3).

Dans notre jeu vidéo d'exemple, nous avons créé un objet pour la soucoupe volante que nous appellerons `ship` et un pour la boule de feu que nous appellerons `fireball`. Le méchant alien portera le nom `alien` et il aura deux costumes : `alienok` lorsqu'il est menaçant et `alienhit` une fois qu'il a été touché par votre boule de feu.

Pour que la suite des opérations soit plus simple, nous conseillons maintenant de repositionner les objets dans la scène : amenez la soucoupe en bas de l'écran au milieu, l'alien tout en haut et la boule de feu à peu près au milieu. Cela correspond aux positions qu'ils auront au cours du jeu.

Contrôler l'exécution des scripts

Nous avons vu dans le Chapitre 9 comment lancer un script en cliquant dans la zone de script. Nous avons besoin de pouvoir faire lancer certains de nos scripts automatiquement lorsque quelque chose de particulier survient, par exemple lorsque le joueur frappe la touche de tir.

C'est l'occasion de découvrir la catégorie de blocs nommée **Contrôle**. Vous y trouvez des blocs qui permettent de lancer l'exécution d'un script en réponse à un événement, comme la collision d'un objet avec un autre ou la frappe d'une touche. Les blocs **Contrôle** vont permettre de définir les règles et les instructions qui vont constituer l'ossature du comportement de votre jeu.

Drapeau vert pour lancer les scripts

Un des blocs **Contrôle** est indispensable pour lancer le jeu en faisant démarrer tous les scripts de façon synchrone pour tous les objets. Vous savez qu'il y a au-dessus de la scène à droite deux boutons : un drapeau vert et un bouton Stop rouge. Le drapeau vert permet de lancer les scripts. Il existe un bloc **Contrôle** permettant de détecter le fait d'avoir cliqué dans ce drapeau. Ce bloc **Contrôle** possède un bord supérieur incurvé, ce qui signifie qu'il n'est pas possible de lui accoler un bloc au-dessus. En revanche, il possède une mortaise inférieure qui permet d'accoler un autre

bloc en dessous. Vous pouvez insérer un script déclenché par ce drapeau vert dans n'importe lequel de vos objets. En cliquant dans le drapeau, vous démarrez d'un seul geste tous les scripts des différents objets en même temps.

À la fin d'une partie, les monstres et les soucoupes peuvent se trouver à n'importe quel endroit. Il faut donc pour redémarrer une partie repositionner chaque objet à sa position de départ. Pour la soucoupe, il faut ramener sa coordonnée X au centre de l'écran en laissant la coordonnée Y indiquer le bas de l'écran. Il faut également réorienter le vaisseau correctement ainsi que faire passer l'objet de la soucoupe au premier plan pour que tous les autres objets soient derrière. Cela permettra de faire apparaître la boule de feu derrière la soucoupe, donnant ainsi un effet plus réaliste qu'en la faisant apparaître devant.

La Figure 10.4 montre le script qu'il faut écrire pour replacer la soucoupe dans les conditions initiales juste après avoir cliqué dans le bouton de lancement (le drapeau vert). Si vous avez créé vos propres objets graphiques, sachez que la position en Y devra éventuellement être un peu plus haute si la taille de votre objet le réclame.

Dès que votre projet comporte plusieurs objets, vous devez toujours faire attention d'avoir bien sélectionné l'objet désiré avant d'ajouter des blocs dans la zone de script. En effet, chaque objet possède une zone de script à lui, et cette zone peut contenir plusieurs scripts. Pour choisir l'objet, vous cliquez dans la liste des objets en bas à droite.

Figure 10.4 : Le bloc de contrôle de démarrage pour l'objet soucoupe nommé ship.

Le bloc Répéter indéfiniment

Les ordinateurs sont des automates ; ils sont donc très doués pour effectuer des tâches répétitives. Les programmes de jeux demandent souvent à la machine de faire sans cesse les mêmes choses jusqu'à ce que le jeu soit terminé.

Une portion de programme qui est répétée s'appelle une *boucle*.

Vous disposez de deux blocs dans la catégorie **Contrôle** pour demander la répétition d'une série de blocs. Le bloc qui s'appelle **Répéter X fois** permet de faire répéter une série d'instructions le nombre de fois indiqué. L'autre bloc se nomme **Répéter indéfiniment**. Il provoque l'exécution d'une série d'instructions jusqu'à la fin du programme.

La forme de ces deux blocs fait penser à une fourche, ce qui montre qu'ils sont prêts à accueillir d'autres blocs dont il faudra répéter l'exécution. Vous remarquez cependant que le bloc de répétition infinie ne comporte pas de mortaise du côté inférieur parce qu'il n'y a aucun intérêt à ajouter un bloc à sa suite. En effet, ce bloc ne serait jamais exécuté puisque par définition vous demandez de répéter indéfiniment l'instruction précédente. Il n'y a rien après l'infini !

Pour piloter latéralement notre soucoupe, nous devons en permanence scruter l'utilisation des deux flèches gauche et droite, et ce jusqu'à la fin du jeu. S'il n'y avait pas de boucle infinie, le script ne consulterait le clavier qu'une fois puis s'arrêterait.

Accédez si nécessaire à la catégorie **Contrôle** de la palette des blocs et faites glisser le bloc intitulé **Répéter indéfiniment** dans la zone de script, à la fin du bloc qui est démarré avec le drapeau vert.

(Facultatif) Lorsque vous découvrez les boucles, nous vous conseillons de tester ce fonctionnement infini. Prenez un bloc de la catégorie **Mouvement** et insérez-le dans la fourche de la boucle. Dans la Figure 10.5, nous avons inséré une instruction de rotation. Cliquez l'icône du drapeau vert en haut à droite pour admirer la rotation de votre soucoupe autour de son axe. Si vous avez effectué ces tests, pensez à supprimer le bloc de mouvement une fois que vous avez fini de tester.

Figure 10.5 : Essai de boucle de répétition infinie avec une instruction de mouvement.

Contrôle d'un objet avec le clavier

Dans notre jeu, le joueur doit pouvoir déplacer la soucoupe vers la gauche et vers la droite avec les deux flèches. Il nous faut donc rédiger une série de blocs répondant au besoin suivant : « Si le joueur appuie sur la touche Flèche gauche, décaler la soucoupe vers la gauche ». Cette série de blocs doit être insérée dans le bloc de répétition infinie pour que Scratch reste à l'écoute du clavier et déplace l'objet en conséquence. Il faut bien sûr créer un deuxième groupe de blocs similaire pour les déplacements vers la droite.

Pour n'exécuter que dans certaines conditions un groupe de blocs, vous utilisez le bloc de la catégorie **Contrôle** qui s'intitule **Si** (*if*). Il correspond au bloc le plus élémentaire d'une instruction conditionnelle et offre le même aspect en fourche que le bloc de répétition. Vous pouvez donc placer d'autres blocs à l'intérieur qui seront ainsi sous son contrôle. Les blocs du bloc **Si** ne seront exécutés que si les conditions sont réunies. Commencez par déposer un bloc **Si** dans la zone de script.

Le principe de création des scripts Scratch consiste à les encliqueter les uns sous les autres. Vous profitez donc des indices visuels que sont les tenons sur le bord supérieur et les mortaises sur le bord inférieur de chaque bloc. Cela vous permet de savoir pour chaque bloc s'il peut suivre ou précéder un autre bloc. Le bloc conditionnel **Si** contient un évidement en forme de losange destiné à recevoir la ou les conditions dans lesquelles le groupe de blocs dépendants sera exécuté. D'autres blocs possèdent ce losange dans les catégories **Opérateurs** et **Capteurs**, et nous les verrons plus loin.

Justement, le bloc dont nous avons besoin pour se mettre à l'écoute du clavier appartient à la catégorie **Capteurs**. Son nom est **Touche pressée**. Il permet au départ de détecter l'appui sur la barre d'espace. Vous pouvez lui faire détecter une autre touche en ouvrant le petit menu pour la choisir. Ici, nous avons besoin de lui faire détecter la flèche gauche du clavier. Déposez ce bloc dans le trou en losange du bloc **Si** dans la zone de script puis choisissez **Flèche gauche** dans son petit menu local.

La Figure 10.6 montre le pavé qui permet de déplacer la soucoupe vers la gauche. Nous avons utilisé un bloc de mouvement pour décaler la position en X de **– 10** unités. Nous en avons profité pour pencher un peu la soucoupe dans la direction de son déplacement. Vous pourriez même changer son costume pour qu'elle se présente autrement pendant ses déplacements à gauche et à droite, ou ajouter d'autres effets visuels ou sonores.

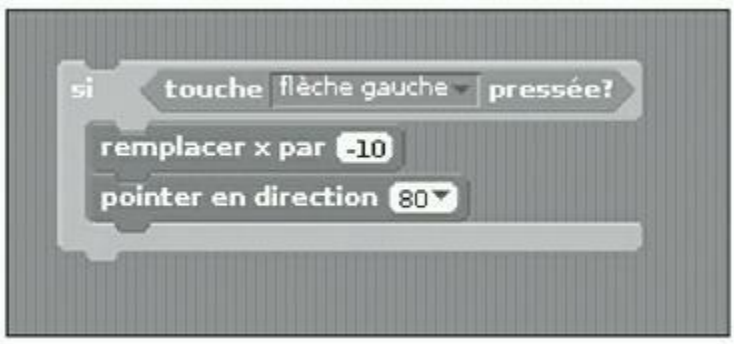


Figure 10.6 : Le bloc conditionnel Si permet de contrôler le décalage de l'objet avec le clavier.

Contrôler un objet avec un autre

Vous disposez souvent de plusieurs manières d'obtenir le même résultat en programmation. C'est le cas du mécanisme de tir de notre jeu. Nous pourrions choisir de détecter l'appui sur la barre d'espace (notre touche de tir) pour lancer un script lié à la boule de feu afin de la faire monter en direction de l'ennemi.

Nous allons pourtant profiter du mécanisme de tir pour vous montrer comment vous pouvez influencer le comportement d'un objet depuis un autre. Il n'est pas possible de déplacer la boule de feu depuis le script de la soucoupe, mais nous pouvons en revanche émettre un message depuis le script de soucoupe pour informer la boule qu'elle doit se déplacer.

Cette solution se présente en deux volets. Tout d'abord, vous allez utiliser un bloc nommé **Envoyer à tous** dans le script de la soucoupe pour qu'elle émette un message en direction de tous les autres objets. Ce message ne

doit être émis que lorsque la barre d'espace est frappée. Nous déposons donc un bloc conditionnel **Si** dans la zone de script de la soucoupe, ajoutons un bloc à losange **Capteur** pour détecter l'appui sur la barre d'espace puis rajoutons un bloc d'émission de message (dans la catégorie **Contrôle**) à l'intérieur de la fourche du bloc conditionnel.

Le bloc **Envoyer à tous** est doté d'un menu local. Ouvrez-le pour choisir **Nouveau** afin de créer un nouveau message. Nous vous conseillons de lui donner le même nom que nous, `fire`.

Cette technique offre plusieurs avantages. Elle permet tout d'abord de rassembler tous les scripts qui contrôlent le jeu dans un seul objet (la soucoupe), ce qui simplifie la lecture du programme. C'est aussi une manière efficace de coordonner le comportement de plusieurs objets. Nous pourrions par exemple donner à notre ennemi un air apeuré dès que nous faisons feu, simplement en changeant son costume. Cela ne demanderait que deux blocs de plus : un bloc **Contrôle** pour détecter la réception du message `fire` et un bloc pour changer de costume. Cette approche est beaucoup plus efficace que celle qui obligerait à créer une boucle pour détecter sans cesse l'appui sur la barre d'espace dans l'objet correspondant à l'ennemi.

Le script complet de la soucoupe est présenté dans la Figure 10.7. Dès que l'appui sur le bouton Drapeau vert de lancement est détecté, la soucoupe est repositionnée puis nous entrons dans une boucle dans laquelle l'objet est déplacé vers la gauche ou vers la droite en fonction des actions sur les flèches. Lorsque c'est la barre d'espace qui est frappée, nous émettons le message `fire` puis nous reprenons la détection des flèches. Vous pouvez lancer ce script pour vérifier que la soucoupe se déplace bien vers la droite et vers la gauche.

Si votre script ne se comporte pas comme prévu, vérifiez-y la forme de vos fourches. Il n'est pas interdit de placer un bloc conditionnel **Si** à l'intérieur d'un autre, mais cela n'est pas ce que nous désirons dans notre exemple, et cela empêchera le jeu de bien détecter les touches. Si vous insérez le groupe de blocs pour la barre d'espace à l'intérieur du bloc de la flèche gauche ou droite, le programme ne testera l'appui sur la barre d'espace que lorsque la flèche correspondante sera pressée.

Passons maintenant à l'objet correspondant à la boule de feu. Cliquez sur l'objet `fireball` dans la liste des objets. Nous allons commencer à y ajouter des scripts. Nous utilisons ici le bloc de contrôle nommé **Quand je reçois fire**. Il va déclencher un script lors de la réception de ce message. Le script est très simple : nous commençons par ramener la boule de feu sous la soucoupe, nous la rendons visible (au départ, elle est cachée par la soucoupe puisqu'elle est derrière), nous lançons la lecture

d’un son de science-fiction choisi parmi les effets standard de Scratch, puis nous faisons glisser l’objet vers le haut de l’écran avant de le cacher.

Dans le bloc de déplacement **Glisser en xx secondes à X : Y :** , nous pouvons déposer un bloc **Position X** à l’endroit où l’on saisit normalement la valeur. Cela permet de conserver pendant tout le déplacement la même position en X, tout en changeant progressivement la position en Y. Le résultat est que la munition va se déplacer verticalement.

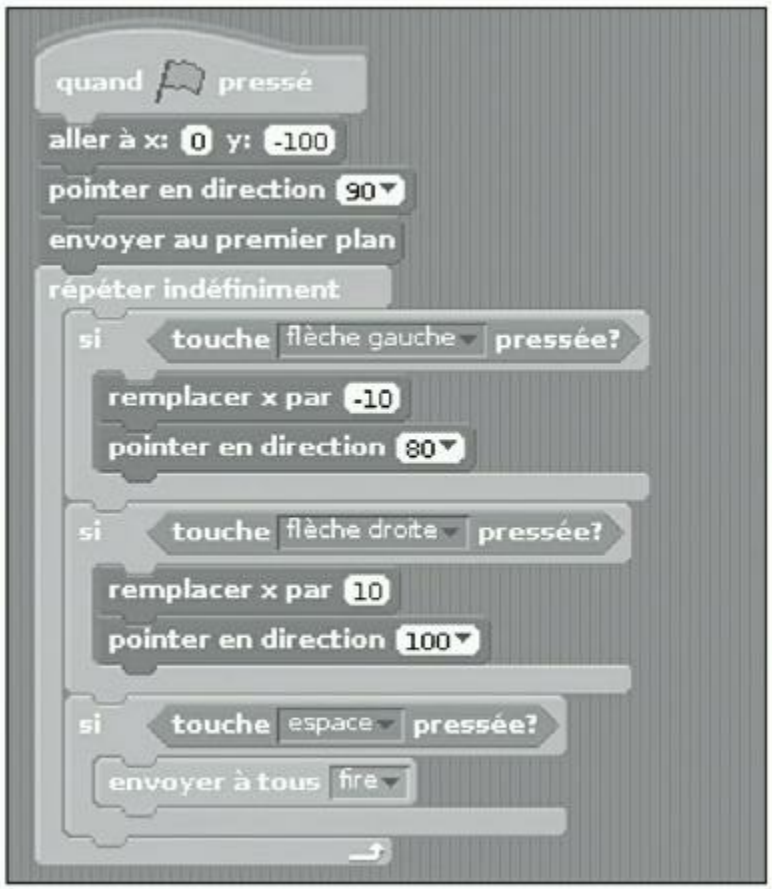

quand pressé
aller à x: 0 y: -100
pointer en direction 90
envoyer au premier plan
répéter indéfiniment
si touche flèche gauche pressée?
remplacer x par -10
pointer en direction 80
si touche flèche droite pressée?
remplacer x par 10
pointer en direction 100
si touche espace pressée?
envoyer à tous fire

Figure 10.7 : Le script de la soucoupe pour réinitialiser puis contrôler les déplacements.

Nous devons ajouter un tout petit script dans la boule de feu pour la cacher au lancement du programme, au cas où elle serait encore visible de la partie précédente.

Rappelons qu’il faut bien faire attention à sélectionner le bon objet avant d’y ajouter un script.

La Figure 10.8 montre les scripts de l’objet `fireball`. Pensez à ajouter le son Laser1 dans le panneau des sons de l’objet avant de créer ce script

(dans le dossier `Electronic`). Testez ensuite le script avec la barre d'espace.

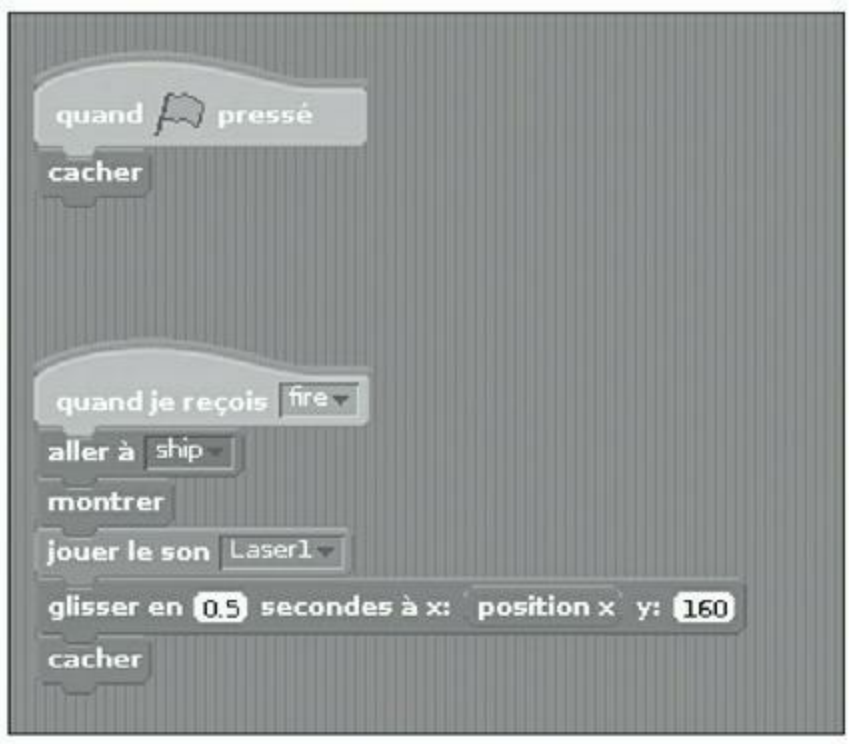


Figure 10.8 : Les deux scripts de l'objet `fireball`.

Exploiter des valeurs aléatoires

Les jeux vidéo deviendraient vite lassants s'ils se comportaient toujours de la même manière. Scratch vous permet de faire générer des nombres au hasard dans vos scripts. Pour que le joueur conserve l'intérêt, nous allons faire apparaître l'ennemi à une position en X variable, en haut d'écran.

Sélectionnez l'objet **alien** dans la liste des objets puis déposez dans sa zone de scripts le bloc de lancement de script avec le drapeau vert. Comme pour les autres objets, nous devons d'abord prévoir un script pour repositionner l'objet en début de jeu. Pour l'alien, il faudrait également prévoir un changement de costume lorsqu'il est touché. Nous devons donc vérifier qu'il apparaît à l'écran dans son costume normal et qu'il est bien visible.

Au départ, l'alien doit apparaître tout en haut de la scène, ce qui correspond à une coordonnée en Y de 150. En ce qui concerne les déplacements latéraux, nous ne voulons pas aller jusqu'au bout des deux côtés parce que l'objet n'est pas très beau à voir lorsqu'il en manque un morceau. Les tests ont montré que la meilleure plage de déplacement

latéral pour l'alien allait de – 180 à 180. Si vous avez créé vos propres objets, il faudra peut-être retoucher un peu ces valeurs.

Déposez dans la zone de script de l'alien un bloc de mouvement **Aller à X : Y :** . Accédez à la catégorie **Opérateurs** de la palette des blocs et repérez le bloc nommé **Nombre aléatoire entre 1 et 10**. Déposez ce bloc dans l'emplacement prévu pour la saisie de la coordonnée X puis modifiez les valeurs de démarrage aléatoire pour qu'elles indiquent **– 180** et **180**.

La Figure 10.9 montre la première version du script de l'alien. Servez-vous du drapeau vert de lancement pour voir s'il place bien l'alien en un point de départ variable tout en haut de l'écran.

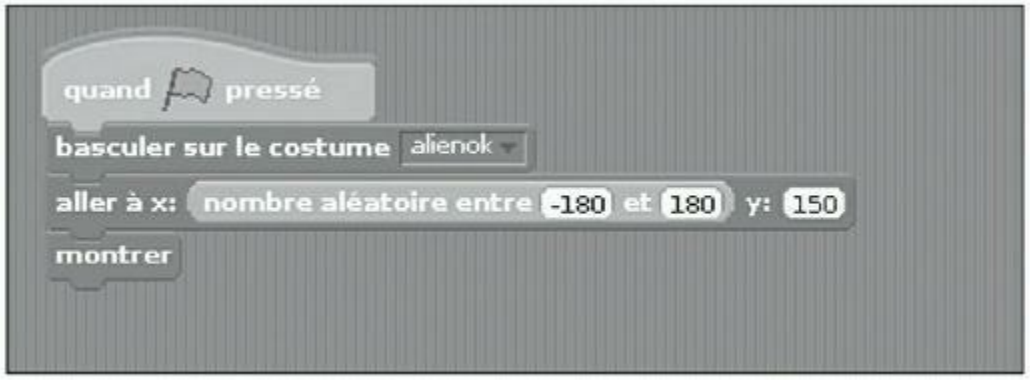


Figure 10.9 : Le script pour positionner l'alien en début de jeu.

Détecter une collision entre deux objets

Il n'y aurait aucun intérêt à tirer sur un alien si cela ne lui faisait ni chaud, ni froid. Pour que le jeu soit intéressant, il faut que l'objet alien réagisse lorsqu'il est touché. Dans de très nombreux jeux, le principe est de faire toucher un objet par un autre (une batte et une balle, une cible et une arme, un poursuivant et un poursuivi). La technique de détection de collision est donc un des savoir-faire fondamentaux dans la conception de jeux vidéo.

Il est possible de détecter que la boule de feu a touché l'objet **alien** dans les scripts de cette boule de feu, mais puisque l'alien doit réagir, c'est dans cet alien que nous allons ajouter notre script.

Pour détecter qu'un objet graphique en touche un autre, nous allons utiliser un bloc de la catégorie **Capteurs**. Nous le combinons avec un bloc

conditionnel **Si** pour déclencher une réaction lorsque l'alien est touché par la boule de feu.

Comme pour la détection des touches dans les scripts de la soucoupe, nous voulons rester en permanence à l'affût d'une collision de l'alien par la boule de feu. Nous ajoutons donc un bloc conditionnel **Si** à l'intérieur d'une boucle de répétition infinie (voir Figure 10.10). Le premier groupe de blocs conditionnels va contenir les instructions de la réaction de l'alien lorsqu'il est touché. Nous changeons son costume pour que l'on sache qu'il est touché, nous lui faisons dire « Aargh », nous faisons jouer un son puis nous cachons l'alien. Après quelques secondes de délai aléatoire, nous repositionnons l'alien en haut d'écran, nous lui restaurons son costume normal et le rendons visible pour que l'horrible cycle d'invasion et de destruction reprenne.

Utilisation de variables

Une variable est un nom dans un programme qui est associé à un emplacement mémoire dans lequel on peut stocker une donnée, la relire et la modifier. Grâce à ce nom (un identifiant), vous pouvez utiliser l'emplacement mémoire dans votre script. Nous avons par exemple besoin de gérer le score d'une partie dans notre jeu, ce que nous allons faire au moyen d'une variable. Cela s'appelle une variable parce que la valeur peut varier au cours du temps. Au départ, le score est à 0 puis il augmente à chaque fois que le joueur touche un ennemi.

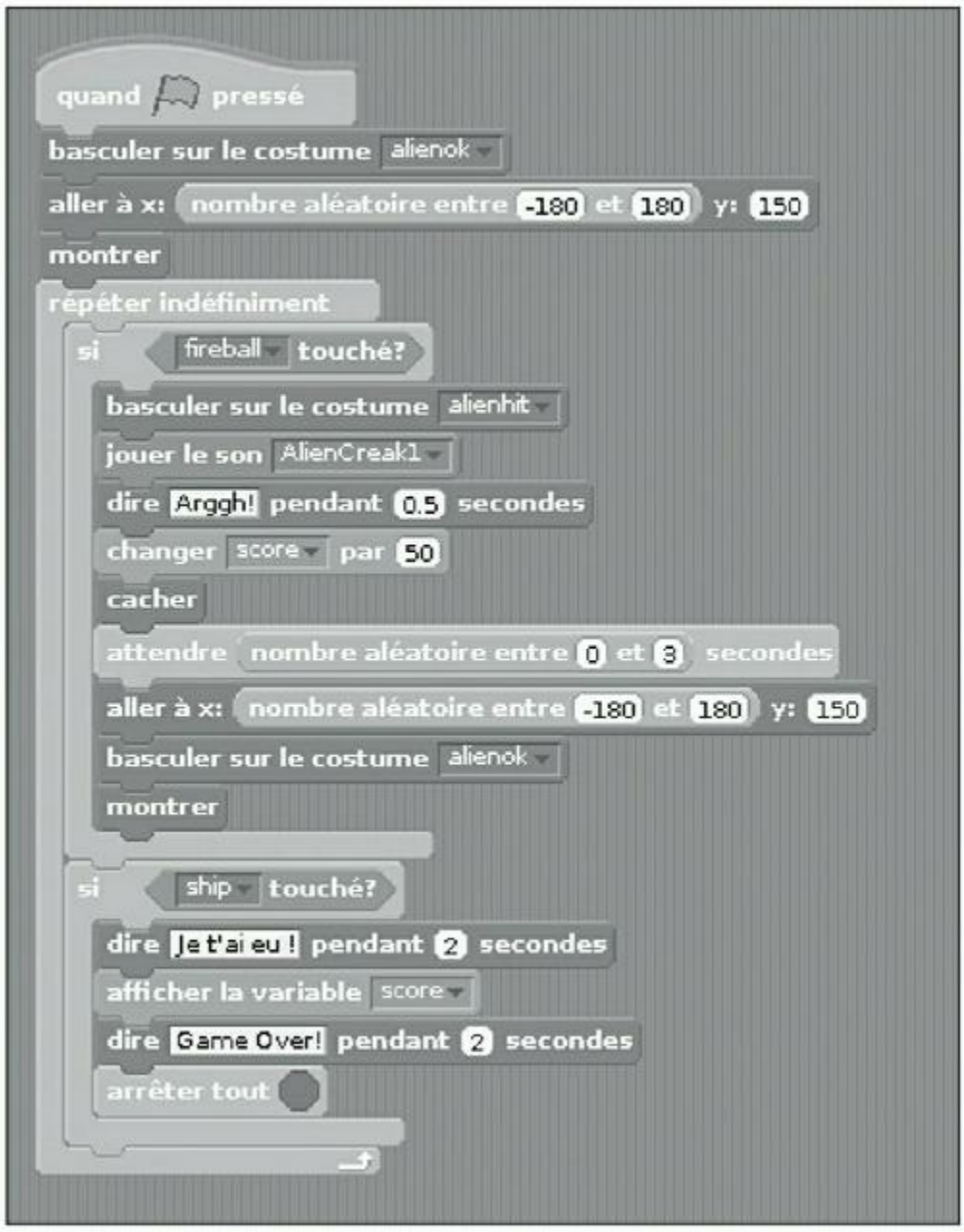

quand pressé
basculer sur le costume alienok
aller à x: nombre aléatoire entre -180 et 180 y: 150
montrer
répéter indéfiniment
si fireball touché?
basculer sur le costume alienhit
jouer le son AlienCreak1
dire Arggh! pendant 0.5 secondes
changer score par 50
cacher
attendre nombre aléatoire entre 0 et 3 secondes
aller à x: nombre aléatoire entre -180 et 180 y: 150
basculer sur le costume alienok
montrer
si ship touché?
dire Je t'ai eu ! pendant 2 secondes
afficher la variable score
dire Game Over! pendant 2 secondes
arrêter tout

Figure 10.10 : Préparation de l'alien et détection de collision.

Nous allons demander à notre script de commencer par remettre le score à 0, pour l'augmenter ensuite à chaque fois qu'un alien est touché, puis nous afficherons le score final en fin de partie. À chaque fois, nous citerons simplement le nom `score`, et le programme saura trouver l'emplacement mémoire correspondant.

Pour créer votre première variable, accédez à la catégorie de blocs **Variables** dans la palette des blocs. Utilisez ensuite le bloc nommé **Nouvelle variable**. Lorsque vous le cliquez, vous devez indiquer le nom choisi pour la variable. Indiquez le nom `score`.

Dans la boîte de saisie du nom de la variable, vous voyez une option en bas qui vous demande si vous voulez que la variable soit utilisable par tous les objets ou seulement par l'objet en cours. Cette option est très

importante. Notre score doit pouvoir être atteint par tous les objets. En revanche, pour une variable qui ne doit servir que pour un objet, mieux vaut choisir l'option correspondante, afin d'empêcher que les autres objets puissent accéder à cette variable qui n'est pas la leur.

De plus, lorsque vous dupliquez un objet, cela duplique également tous ses scripts et toutes ses variables. Si plusieurs objets utilisent des variables homonymes, vous risquez de remarquer des comportements étranges. Les variables locales doivent être indépendantes. Nous en verrons un exemple un peu plus loin lorsque nous créerons un deuxième ennemi.

Dès que vous créez une nouvelle variable, vous voyez apparaître de nouveaux blocs dans la palette des blocs qui vont permettre de modifier la valeur de la variable et de décider de la cacher ou de la rendre visible sur la scène. Notre score doit augmenter de 50 à chaque toucher (soyons généreux, ce n'est pas un jeu très facile !). Vous pouvez donc déposer le bloc **Changer score par 1** dans le script puis lui donner la valeur **50**. Ce bloc doit être placé dans la fourche du bloc conditionnel **Si** qui permet de détecter la collision entre un alien et la boule de feu (revoyez la Figure 10.10).

Nous avons vu dans le Chapitre 9 que nous pouvions afficher la position et l'orientation d'un objet sur la scène. Les valeurs des variables sont par défaut affichées elles aussi, au départ dans le coin supérieur gauche. Vous pouvez les déplacer. Cet affichage est très pratique pour trouver les causes des erreurs, mais cela ralentit beaucoup le fonctionnement. Nous vous conseillons donc de désélectionner la case à cocher de la nouvelle variable `score` dans la palette des blocs pour qu'elle ne soit plus affichée sur la scène.

Lorsque le jeu sera terminé, l'ennemi devra descendre par palier dans l'écran en direction de la soucoupe. La partie est perdue si l'ennemi touche la soucoupe du joueur. À ce moment, nous voulons afficher la valeur de la variable `score` sur la scène puis nous servir d'un bloc de contrôle pour arrêter l'exécution de tous les scripts et donc provoquer la fin du programme. La Figure 10.10 précédente contient déjà tous les blocs permettant d'obtenir ce résultat. La structure est à peu près la même que celle des blocs qui servent à détecter qu'un ennemi est touché.

Déplacer automatiquement un objet

Pour l'instant, le déplacement de notre alien n'est pas satisfaisant, mais cela simplifie la suite de la programmation et les tests. Nous disposons

maintenant d'une soucoupe que le joueur peut contrôler, d'un mécanisme de tir et d'un alien qui succombe puis se régénère. Tout cela peut être testé et nous pouvons corriger les problèmes sans avoir à gérer les attaques de l'alien.

Notre alien se déplace de gauche à droite puis de droite à gauche. À chaque changement de sens, il doit descendre un peu pour se rapprocher de la soucoupe. C'est un comportement un peu plus complexe, mais nous allons pouvoir le programmer presque uniquement avec les blocs que nous connaissons déjà. L'art de la programmation consiste aussi à trouver la bonne manière d'utiliser les différentes commandes disponibles pour atteindre un certain objectif.

Il nous faut d'abord créer une nouvelle variable que nous allons appeler `leapsize` (largeur de pas). Lors de chaque tour de la boucle infinie de l'alien, nous déplaçons l'objet puis testons s'il a touché une boule de feu ou la soucoupe. La variable `leapsize` contient la quantité de déplacement pour le prochain pas dans le sens des X. Lorsque l'alien va à droite, `leapsize` vaut 20 et lorsqu'il va à gauche, la variable vaut – 20. Revoyez si nécessaire la Figure 9.4.

Comme pour la variable précédente, vous devez choisir lors de la création de la variable `leapsize` si elle doit être utilisable par tous les objets ou seulement par l'objet courant (l'alien). Pensez à choisir l'option **Seulement pour cet objet**. Si vous ne le faites pas, vous aurez des problèmes lorsque vous créerez une copie de cet objet parce que vous aurez deux objets alien qui vont utiliser la même variable `leapsize`. Cette variable doit être unique pour chaque objet et sa valeur doit dépendre à tout moment de la position actuelle de l'objet à l'écran. Si vous constatez que des objets se trouvent bloqués sur un bord de l'écran, c'est sans doute parce qu'ils utilisent une variable qui n'est pas la leur.

Dès que l'alien rencontre un bord de l'écran, nous modifions la valeur de la variable `leapsize` pour que l'alien reprenne dans l'autre sens, à raison de 20 unités par pas. De plus, nous en profitons pour le faire descendre de 20 unités en direction de la soucoupe (dans le sens des Y).

Le groupe de blocs pour les déplacements est présenté dans la Figure 10.11. Tout ce groupe doit être inséré dans la boucle infinie du script de l'alien, au début de cette boucle.

Les blocs de couleur verte **Opérateurs** vous permettent d'écrire des instructions encore plus complexes pour faire des additions, tester l'égalité d'une valeur et d'une autre et même combiner plusieurs conditions dans un seul bloc **Si**. Cela dit, du fait qu'il est possible

d’imbriquer des blocs dans des blocs, la lecture peut devenir quelque peu ardue.

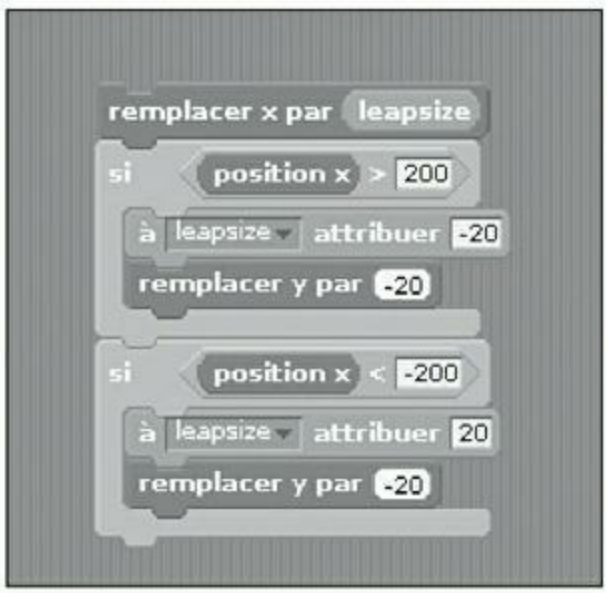

remplacer x par leapsize
si position x > 200
à leapsize attribuer -20
remplacer y par -20
si position x < -200
à leapsize attribuer 20
remplacer y par -20

Figure 10.11 : L’extrait de script pour les déplacements de l’objet alien.

Les blocs conditionnels **Si** de notre script de déplacement de l’alien se servent de blocs de la catégorie **Opérateurs** pour comparer la position actuelle en X avec une valeur afin de savoir quand l’objet touche un des deux bords. Nos essais ont montré que les valeurs – 200 et 200 correspondaient aux limites en X pour éviter que l’objet disparaisse de la vue sur la scène. Vu que les blocs de comparaison ont une forme de losange, nous pouvons en déposer deux dans l’évidement du bloc conditionnel **Si**. Un bloc va servir à tester si la position X est supérieure à **200** et un autre si elle est inférieure à **– 200**. (Nous ne pouvons pas tester une égalité avec 200 et – 200 parce que l’objet commence à une position aléatoire puis augmente de 20 unités. S’il démarre par exemple à 170, il sautera de 190 à 210 sans jamais être égal à 200.)

Il nous reste à ajouter tout au début du script un bloc pour attribuer au départ à la variable `leapsize` la valeur 20 pour que l’alien bouge. Dans la palette des blocs, déposez le bloc **À leapsize attribuer X** au début du script, juste après **Quand drapeau vert pressé** puis changez la valeur pour qu’elle indique **20**. Le bloc doit être situé avant la boucle de répétition infinie.

Détecter et corriger un bogue

Dans les projets de création de logiciels commerciaux, beaucoup de temps et d'argent sont dépensés à tester les programmes pour garantir qu'ils fonctionnent comme prévu et pour corriger ce qui ne va pas. Les erreurs dans les programmes sont en général appelées des bogues (*bugs*). Notre petit jeu vidéo comporte un bogue qui permet au joueur de tricher.

En effet, pendant que la boule de feu se déplace vers le haut de l'écran, si le joueur fait à nouveau feu, cela lance une nouvelle boule. La boule qui a été lancée disparaît et la nouvelle repart de la soucoupe. Cela permet au joueur de tirer à répétition sans être pénalisé s'il a mal visé.

Nous allons ajouter une variable pour savoir si nous venons de tirer afin d'interdire un nouveau tir tant que la boule de feu n'a pas terminé sa course. Une telle variable, dont le seul but est de savoir si quelque chose est en cours ou pas, se nomme un drapeau (*flag*). Notre drapeau de tir doit nous permettre de savoir si la boule a été tirée. Il y a donc deux valeurs possibles. Lorsque la boule se déplace, nous donnons au drapeau la valeur **1** et nous la remettons à **0** quand il n'y a pas de boule de feu à l'écran.

Dans la palette des blocs, accédez à la catégorie **Variables** et cliquez l'option de création de variable. Donnez-lui le nom `firingflag` et vérifiez que l'option sélectionnée rende cette variable accessible à tous les objets.

Une fois la variable créée, nous pouvons déposer un bloc d'attribution de valeur pour cette variable au début de la partie du script de la boule de feu correspondant au tir. Donnez à cette variable la valeur **1** à cet endroit. Mettez en place un deuxième exemplaire du bloc d'attribution de valeur à la fin du script en donnant cette fois-ci la valeur **0**. Il reste à mettre en place un deuxième exemplaire de ce bloc d'attribution avec la valeur **0** tout au début du script, pour tenir compte du cas où vous avez été touché alors que la boule de feu était encore visible. Le script définitif de la boule de feu est présenté dans la Figure 10.12.

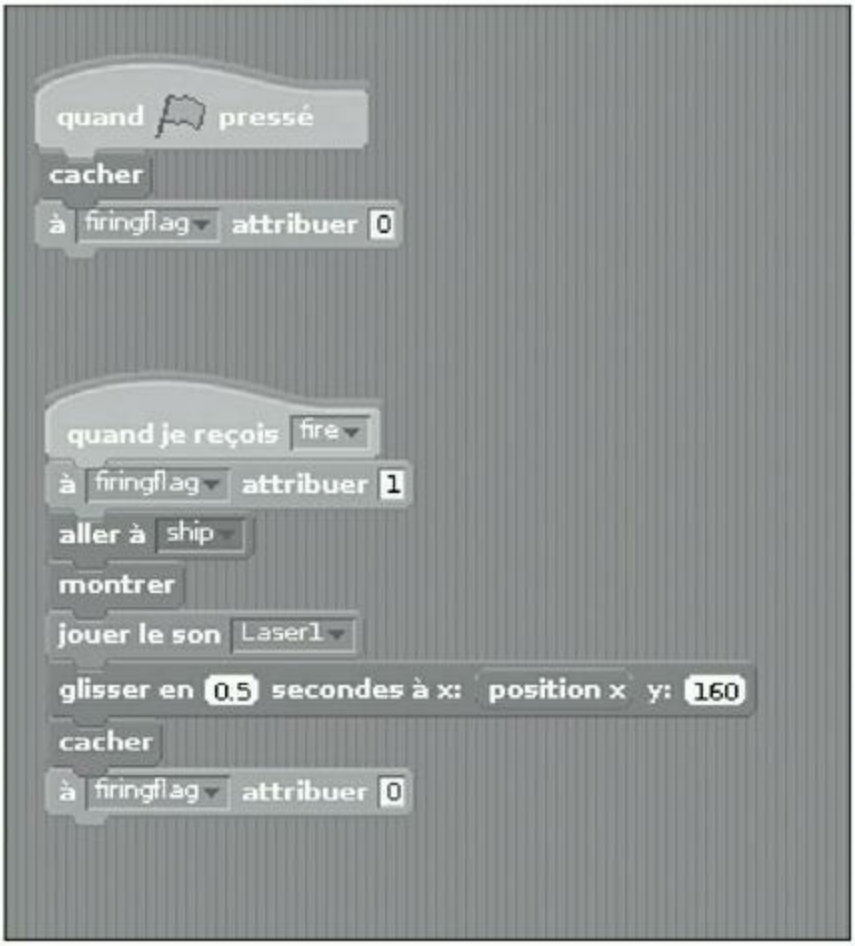


Figure 10.12 : Le script complet de la boule de feu avec le drapeau de tir.

Il nous reste à intervenir dans le script de la soucoupe pour qu'elle tire profit de cette variable. Le tir ne doit être possible que si la variable `firingflag` vaut **0**. Cette partie est un peu plus complexe, parce qu'il va nous falloir imbriquer plusieurs blocs différents.

Sélectionnez l'objet `ship` pour afficher ses scripts. Le bloc qu'il nous faut modifier est le bloc conditionnel **Si** qui teste l'appui sur la barre d'espace. La Figure 10.13 montre les différentes étapes d'un bloc conditionnel double. Ces cinq blocs ont été déposés en guise d'exemple, à l'écart du reste du script.

Commencez par vider le trou en losange du bloc conditionnel **Si**. Dans cet endroit libéré, déposez un bloc pour l'opérateur à double condition **Et**. Dorénavant, les actions contrôlées par le bloc **Si** ne seront exécutées que si les deux conditions de part et d'autre du **Et** sont remplies. La première condition reste l'appui sur la barre d'espace. Vous pouvez donc remettre en place le bloc que vous avez déposé à l'écart dans le premier losange à gauche du **Et**. La seconde condition est que `firingflag` soit égale à **0**. Déposez le bloc de l'opérateur **'=0'** dans le losange à droite du **Et** puis

déposez le nom de la variable `firingflag` dans la case vide de gauche de ce test d'égalité.

Dorénavant, vous ne pourrez tirer qu'une munition à la fois. Vos ennemis sont des aliens, mais ils méritent un combat loyal !

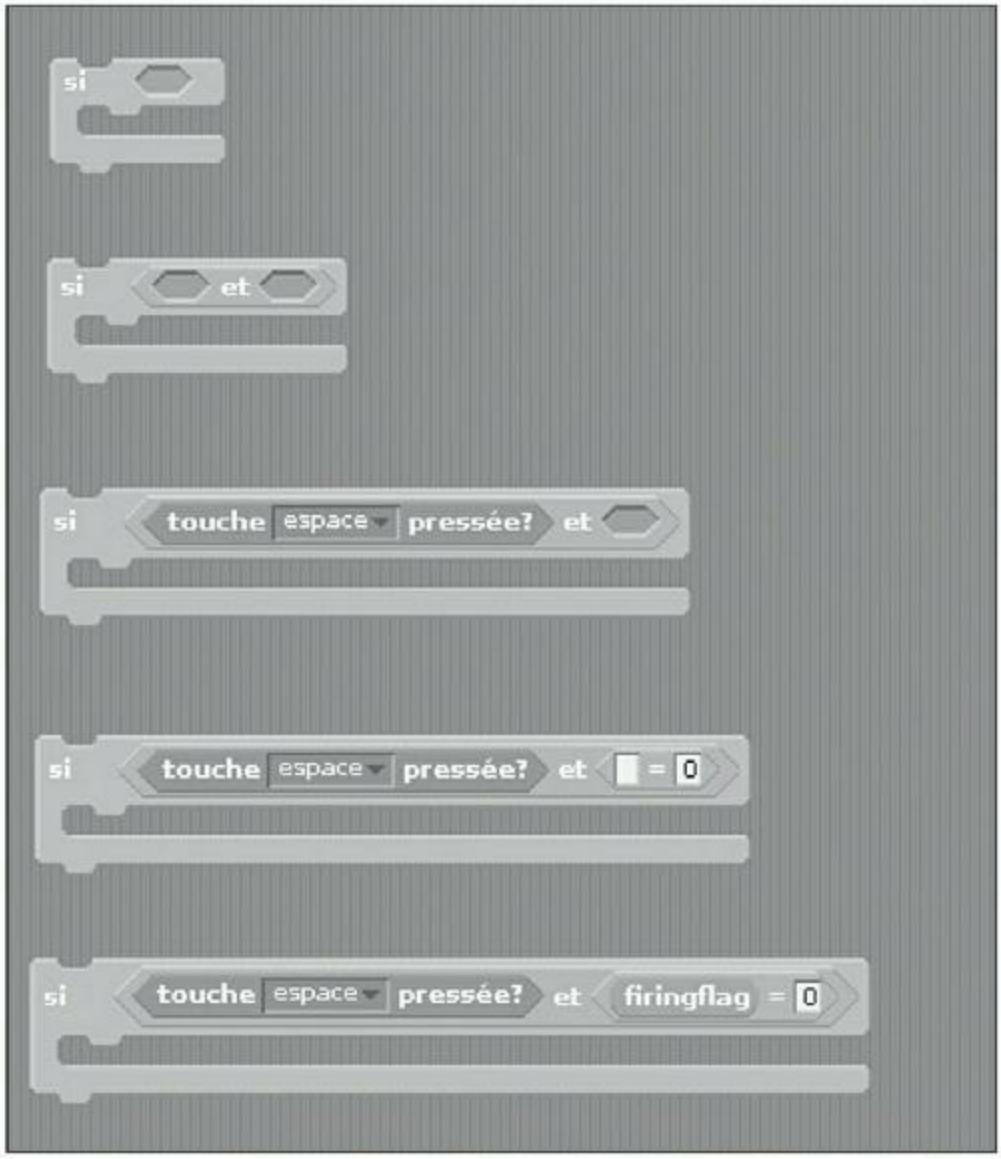


Figure 10.13 : Les différentes étapes de construction d'un bloc conditionnel utilisant deux conditions.

Ajouter un script pour la scène

Vous pouvez non seulement associer des scripts à vos objets graphiques, mais également à la scène. Dans la liste des objets, cliquez à gauche sur l'icône **Scène**. Vous constatez qu'elle offre une zone de script. Lorsque vous avez besoin de faire une retouche sur un script, il est parfois fastidieux de passer en revue les zones de scripts de tous les objets pour trouver où a été placé un bloc. Les actions qui concernent la totalité du jeu

sans dépendre d'un objet en particulier seront bien mieux réunies dans la zone de script de la scène.

Dans notre jeu, il nous suffit d'ajouter un bloc au niveau de la scène pour remettre le score à 0 dès que nous lançons le jeu par le drapeau vert. Si nous oublions cette remise à 0, le score va atteindre des sommets stratosphériques, malgré le redémarrage du jeu. Vous pouvez aussi ajouter un bloc pour cacher le score en début de partie. Il sera réaffiché par l'alien en fin de jeu. La Figure 10.14 montre cette technique.

Figure 10.14 : Les scripts de la scène.

Dupliquer un objet

Nous avons créé notre objet alien en prenant soin de définir la variable `leapsize` comme variable locale de cet objet. C'est grâce à cette précaution que nous pouvons très simplement créer un deuxième ennemi pour corser le jeu. Cliquez droit sur l'objet **alien** dans la liste des objets et choisissez **Dupliquer**. Dorénavant, vous devez affronter deux ennemis à la fois.

Tester le jeu

Pour tester votre jeu en profitant de toute la surface de l'écran, utilisez l'icône représentant un chevalet dans l'angle supérieur droit de la fenêtre (**Passer en mode présentation**). La scène prend toute la surface de l'écran et vous pouvez lancer le jeu avec le drapeau vert. Pour revenir à l'ancien format d'affichage, utilisez la flèche dans l'angle supérieur gauche. La Figure 10.15 montre un aperçu du jeu, mais le vôtre pourra être différent si vous avez choisi d'autres objets graphiques.

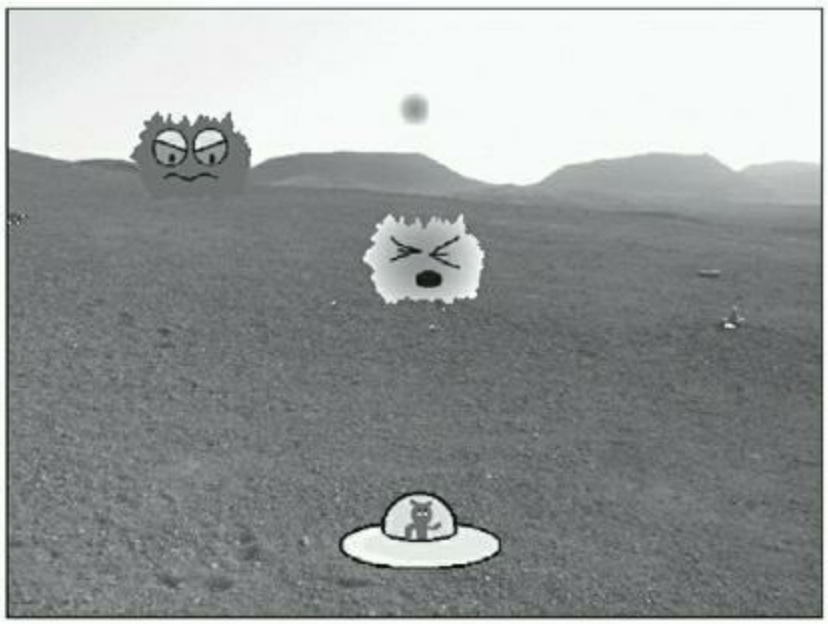


Figure 10.15 : Et un de moins ! Le jeu est terminé.

Régler la vitesse de jeu

Sur le Raspberry Pi de l'auteur, le jeu tourne à une vitesse qui reste jouable, mais un des créateurs du Raspberry Pi, Eben Upton, a annoncé qu'une de ses priorités était d'augmenter les performances de Scratch. De ce fait, si le jeu devient trop rapide sur votre Raspberry Pi avec la nouvelle version du logiciel, sachez que vous pouvez ralentir un peu l'ardeur des aliens en réduisant la valeur de la variable `leapsize` (à trois endroits). Vous pouvez également réduire la quantité de leur décalage vers le bas (en Y) à chaque changement de sens. Vous pouvez enfin ajouter une petite pause dans la boucle de l'alien, mais cela va réduire la précision de la détection des collisions.

Vous ralentissez aussi le jeu en ajoutant plus d'objets et d'aliens. C'est une façon intelligente de tirer profit du supplément de puissance des cartes récentes.

Pour aller plus loin

Ce chapitre nous a permis de présenter de nombreux concepts essentiels de programmation : les boucles de répétition, les opérateurs, les conditions et les variables. Vous savez maintenant comment utiliser Scratch pour concevoir vos propres jeux avec des objets qui agissent et réagissent aux actions du joueur. Vous pouvez commencer par personnaliser ce jeu d'exemple en dessinant vos propres objets, en faisant

varier la vitesse des aliens dès qu'ils sont touchés ou leur mode de déplacement. Cela dit, la grande aventure qui vous attend consiste à vous servir de ce que vous avez appris dans ce chapitre, en y ajoutant quelques autres blocs non présentés ici pour créer votre premier jeu vraiment personnel.

Pour en savoir plus au sujet de Scratch et pour trouver de nombreux jeux et animations que d'autres personnes ont mis à disposition, allez voir le site Web `http://scratch.mit.edu`. Vous pourrez publier vos propres œuvres et ainsi obtenir l'avis des autres passionnés de Scratch.

Voyez aussi un livre récemment écrit par l'un des coauteurs, Sean McManus : *Code ton jeu vidéo* aux éditions Fleurus.

Chapitre 11

Programmez en Python

DANS CE CHAPITRE :

- **Saisie de données utilisateur et affichage**
- **Variables, chaînes, listes et dictionnaires**
- **Boucles de répétition for et while**
- **Instructions conditionnelles pour décider**
- **Définition et appel d'une fonction**

Ce chapitre vous propose une découverte du langage Python, qui est un langage de programmation très puissant utilisé pour produire des logiciels commerciaux.

Une des méthodes les plus pratiques pour découvrir la programmation consiste à étudier les programmes déjà écrits par d'autres. Dans ce chapitre, nous allons vous guider pas à pas dans deux programmes. Le premier est un programme très simple pour calculer et afficher une table de multiplication. L'autre est un petit simulateur d'intelligence qui va vous permettre de discuter avec votre Raspberry.

Votre apprentissage sera plus efficace si vous saisissez le code source des exemples selon nos conseils, mais vous pouvez également télécharger les programmes source qui font partie du fichier d'archive accompagnant le livre. Pour savoir comment récupérer ce fichier, voyez l'introduction.

L'espace disponible dans ce livre pour parler du langage Python ne permet pas de décrire ce langage en détail, des livres entiers y étant consacrés. Ce chapitre va vous permettre de faire vos premiers pas. En découvrant le fonctionnement de nos deux exemples, vous apprendrez certains des principes de Python et de la programmation et vous saurez comment construire un programme en Python.

Ces nouvelles compétences vont vous servir pour réaliser les projets d'électronique de la Partie 5 et le jeu vidéo du Chapitre 12 basé sur Pygame Zero.

Certaines lignes de code sont trop longues par rapport à la largeur disponible dans ce livre. Les lignes concernées comportent un petit symbole. Quand vous voyez ce symbole, vous devez continuer la saisie sans forcer un retour à la ligne !

Démarrer Python

Dès le départ, votre Raspberry Pi offre deux versions de Python : Python 2.7 et Python 3. En général, lorsqu'un logiciel est mis à jour, la nouvelle version reste compatible avec l'ancienne et la remplace. Il en va différemment pour Python. Python 3 a été rendu non compatible dès le départ, en raison de ses nombreuses évolutions. Les programmes écrits pour Python 2.7 ne fonctionnent pas directement avec Python 3, et *vice versa*. Dans ce livre, nous choisissons d'utiliser la nouvelle version Python 3.

Sachez que la différence la plus visible est que la commande `print` est devenue une fonction qui s'écrit donc `print()`. Pour en savoir plus, les anglophones liront la page suivante : `http://docs.python.org/3.0/whatsnew/3.0.html`.

Dès que possible, les programmeurs préfèrent utiliser un atelier de développement, appelé également Environnement de Développement Intégré ou IDE (*Integrated Development Environment*). Il s'agit d'un outil se présentant dans une fenêtre et rassemblant tous les éléments permettant de créer et de tester les programmes. Pour Python, l'atelier s'appelle IDLE. Il est offert en deux versions : *IDLE* (pour Python 2.7) et *IDLE 3* (pour Python 3). Vous trouverez les deux icônes sur votre bureau, mais ne cliquez aucune des deux.

En effet, un nouvel outil de création Python est apparu récemment. Il se nomme **Thonny**. Son utilisation est encore plus intuitive que celle d'IDLE, surtout du fait qu'il regroupe les actions de rédaction et de test dans une seule fenêtre. Nous utiliserons Thonny dans ce projet, mais les programmes peuvent être produits tout aussi bien avec IDLE 3.

Pour démarrer Thonny, ouvrez le menu général puis le menu **Programmation** et cliquez **Thonny Python IDE**.

Saisie de commandes Python

Au démarrage de Thonny, vous voyez apparaître une fenêtre avec deux panneaux vides (Figure 11.1).

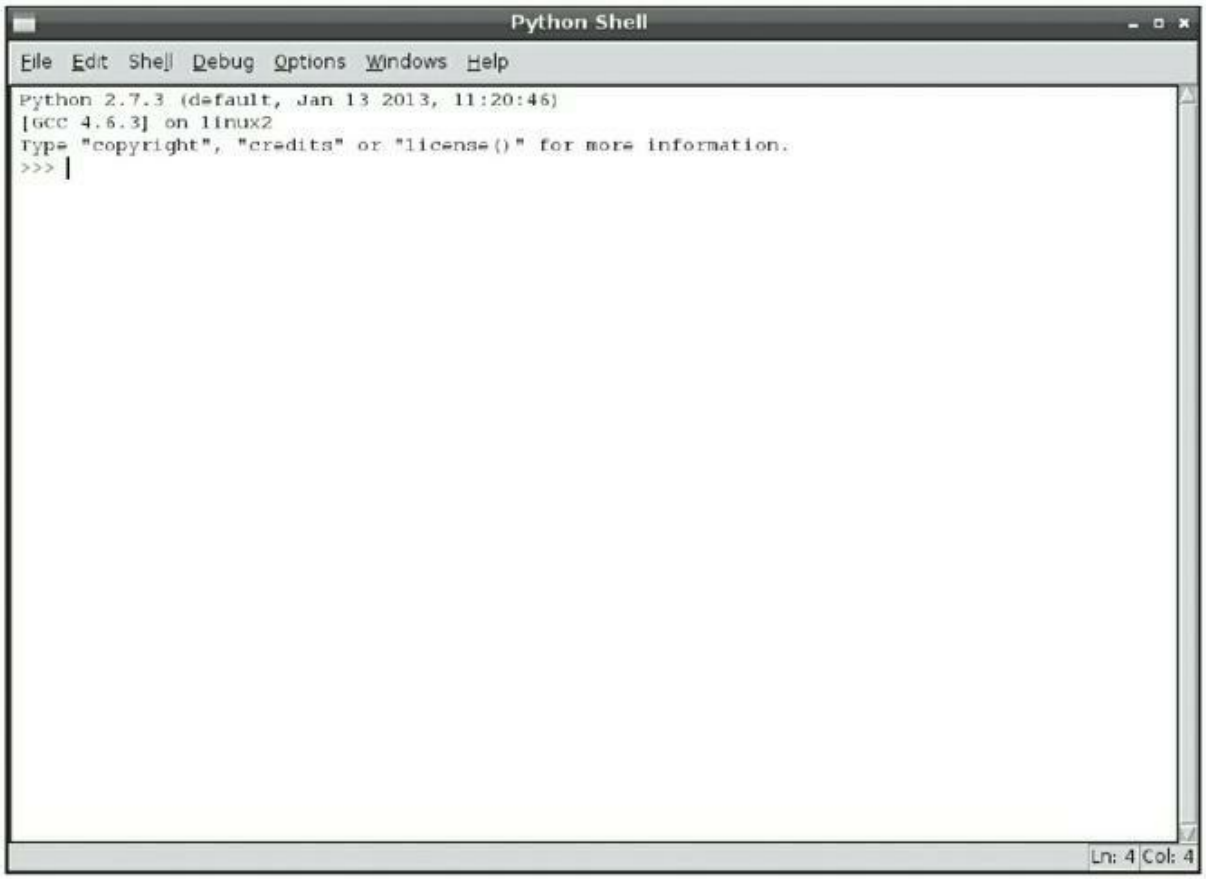

Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for more information.
>>>
Ln: 4 Col: 4

Figure 11.1 : L'éditeur Thonny avec son panneau inférieur Python Shell.

Le panneau supérieur est celui de l'éditeur de texte. C'est ici que vous allez saisir les instructions en langage Python qui vont constituer vos programmes.

Le panneau inférieur donne accès à l'interpréteur Python (shell). Vous remarquez les trois chevrons en début de la dernière ligne de texte ; elle représente l'invite de l'interpréteur de commandes (*shell*) et elle signifie que Python est prêt à accepter une commande que vous pouvez saisir sur cette ligne. Vérifiez cela en saisissant la commande `licence()` qui affiche l'historique de Python puis les conditions d'utilisation. Si vous n'avez pas le temps de lire tous ces détails juridiques, abandonnez la suite en frappant la touche **Q** puis validez par **Entrée**.

Vous pouvez augmenter la hauteur du panneau inférieur en déplaçant la barre de réglage qui se trouve à la frontière entre les deux panneaux. Utilisez aussi le bouton d'agrandissement de la fenêtre de Thonny dans

son coin supérieur droit.

Une des commandes les plus élémentaires dans tous les langages de programmation est celle qui demande à l'ordinateur d'afficher un message de bienvenue à l'écran. Dans Python (et dans quelques autres langages), la commande (c'est plus exactement une fonction) porte le nom `print()` et vous l'utilisez ainsi. Ne saisissez que ce qui est en gras puis validez par la touche Entrée :

```
>>> print("Salut")
Salut
>>>
```

Les parenthèses englobent le paramètre de la fonction, c'est-à-dire ce qu'elle doit afficher. Les guillemets sont indispensables pour délimiter le ou les mots qui constituent le contenu du message.

Pendant la saisie, vous avez remarqué que Thonny ajoutait des couleurs au texte en fonction de ce qu'il croit comprendre : du gris quand il manque la parenthèse fermante et du vert pour du texte à afficher. Cette coloration syntaxique est d'un grand secours pour les débutants en Python. Ne validez pas votre saisie par Entrée tant que la barre de surveillance est présente : elle signale que la saisie n'est pas valable.

Ce que vous saisissez entre guillemets après le nom `print (` sera affiché tel quel puis Python réaffichera les trois chevrons de son invite pour vous proposer de saisir une autre commande.

Comme l'interpréteur shell du système Linux (vu dans le Chapitre 5), le langage Python distingue la casse, et ne confond donc pas les majuscules et les minuscules. Vous devez saisir le nom de commande `print` en minuscules. Dans le cas contraire, Python affichera une erreur de syntaxe faute de comprendre le nom de la commande. En revanche, vous pouvez bien sûr écrire comme vous le désirez le texte entre les guillemets. Il sera affiché tel quel. Voici trois exemples dont les deux premiers renvoient une erreur :

```
>>> PRINT("salut Corinne!")
SyntaxError: invalid syntax
>>> Print("salut Corinne!")
SyntaxError: invalid syntax
>>> print("salut Corinne!")
salut Corinne!
```

Si vous faites une faute de saisie, vous revenez à l'invite de départ avec la combinaison de touches Ctrl + C. C'est notamment utile dans la saisie de certains signes (parenthèses, crochets), car l'éditeur peut croire que vous voulez saisir une instruction longue sur plusieurs lignes et donc ne pas faire revenir à la marge ou à l'invite quand vous frappez Entrée. Dans tous les cas, la solution, c'est Ctrl + C.

Quelques calculs en Python

Vous pouvez effectuer des opérations arithmétiques dans l'interpréteur. Le Tableau 11.1 dresse la liste des opérateurs mathématiques disponibles. Vous indiquez l'opérateur et les valeurs après le nom de commande `print`, comme ceci :

```
>>> print(5+5)
10
>>> print(9-4)
5
>>> print(7*7)
49
>>> print(10/2)
5
```

Vous remarquez que nous n'avons pas ajouté de guillemets dans ces exemples. Que se passerait-il si vous en aviez ajouté ? Python aurait affiché littéralement ce que vous aviez demandé d'afficher :

```
>>> print("5+5")
5+5
```

Lorsque vous devez ignorer la partie fractionnaire et effectuer un arrondi, vous utilisez l'opérateur de division avec arrondi `//`. Comparez ces deux essais :

```
>>> print(10.0 / 3)
3.33333333333
>>> print(10.0 // 3)
3.0
```

Un opérateur que vous n'avez peut-être pas encore rencontré est le modulo qui correspond au symbole %. Il permet de récupérer le reste d'une division. Voici deux exemples :

```
>>> print(10 % 3)
1
>>> print(10 % 2)
0
```

Cet opérateur vous permet notamment de savoir si un nombre est divisible par un autre (auquel cas son modulo reste est égal à 0).

Tableau 11.1 : Opérateurs mathématiques de Python.

Opérateur	Description
+	Addition
–	Soustraction
*	Multiplication
/	Division
//	Division, avec arrondi
%	Modulo, pour récupérer le reste d'une division

Les calculs que nous venons d'essayer sont vraiment élémentaires, mais vous pouvez en exprimer de plus complexes en combinant les chiffres et opérateurs avec des parenthèses. Comme en algèbre, le contenu des parenthèses les plus intérieures est calculé d'abord. Voici deux exemples :

```
>>> print( (10.0 / 3) * 2 )
6.66666666667
>>> print( 10.0 / (3 * 2) )
1.66666666667
```

Vous pouvez même faire un peu de mathématiques dans l'interpréteur en saisissant directement vos formules sans la commande `print`, mais cette commande est indispensable pour créer des programmes, comme nous allons le voir maintenant.

Les espaces ajoutées autour des opérateurs et opérandes sont facultatives, mais elles améliorent la lisibilité et rendent visibles les sous-opérations.

Un programme de tables de multiplication

Nous allons maintenant créer notre premier vrai programme. Il va calculer une table de multiplication. Nous invitons l'utilisateur à saisir un chiffre, par exemple **7**, puis le programme va afficher les différents multiples 7, 14, 21, *etc*. Ce programme est très bref, mais il suffit à montrer comment créer un programme, comment utiliser des variables pour stocker des valeurs numériques, comment demander à l'utilisateur de saisir une donnée et comment créer un bloc de programme qui se répète (une boucle). Nous allons partir de ce que nous savons faire avec la commande **print()**. Si vous avez lu et expérimenté avec les Chapitres 9 et 10 consacrés à Scratch, vous allez retrouver certaines idées déjà présentées.

Rédiger et tester un premier programme Python

Lorsque vous saisissez des instructions sur la ligne de commande de l'interpréteur, vous êtes forcé de les ressaisir à chaque fois et vous ne pouvez travailler qu'une instruction à la fois. Votre commande est exécutée immédiatement, mais cela empêche de faire des choses un tant soit peu sophistiquées. La solution à ces limitations consiste à créer un programme, c'est-à-dire une série de lignes d'instructions qui constitue un code source et que vous pouvez enregistrer dans un fichier puis recharger.

Pour créer votre premier programme, vous basculez dans le panneau supérieur de Thonny. Au départ, c'est une fenêtre vide dont l'onglet indique `<untitled>`.

Lorsque vous saisissez une commande en mode script, elle n'est pas exécutée lorsque vous passez à la ligne suivante. La fenêtre est un éditeur de texte ; vous pouvez saisir toute la série de commandes qui vont constituer votre programme. C'est vous qui décidez quand vous voudrez lancer l'exécution de la série de commandes.

Commencez par saisir les trois lignes suivantes en mode script :

```
# Calcul de tables de multiplication
```

```
print("Ce programme calcule les tables")
print("de multiplication sur Raspberry Pi")
```

La fenêtre doit ressembler à celle de la Figure 11.2. Les deux commandes `print()` vous sont déjà connues, mais la première ligne est nouvelle. En langage Python, tout ce qui suit le signe # est ignoré par l'ordinateur, car c'est un commentaire. Vous pouvez ainsi ajouter des informations destinées à ceux qui vont lire le code source. Les bons programmeurs rédigent le code source de façon suffisamment lisible pour qu'il soit compréhensible sans commentaires, mais il est toujours conseillé d'ajouter quelques petits messages par-ci par-là pour vous remémorer plus tard ce que fait votre programme et pour aider les autres à le comprendre. Nous avons prévu un commentaire sur une ligne tout au début du programme, comme c'est de coutume, ce qui nous permettra de nous remémorer son but lorsque nous le rouvrirons plus tard.

Pour enregistrer notre programme source dans son état actuel, ouvrez le menu **File** (Fichier) en haut de la fenêtre de Thonny et choisissez **Save** (Enregistrer). C'est dans ce même menu que vous pourrez recharger le fichier.

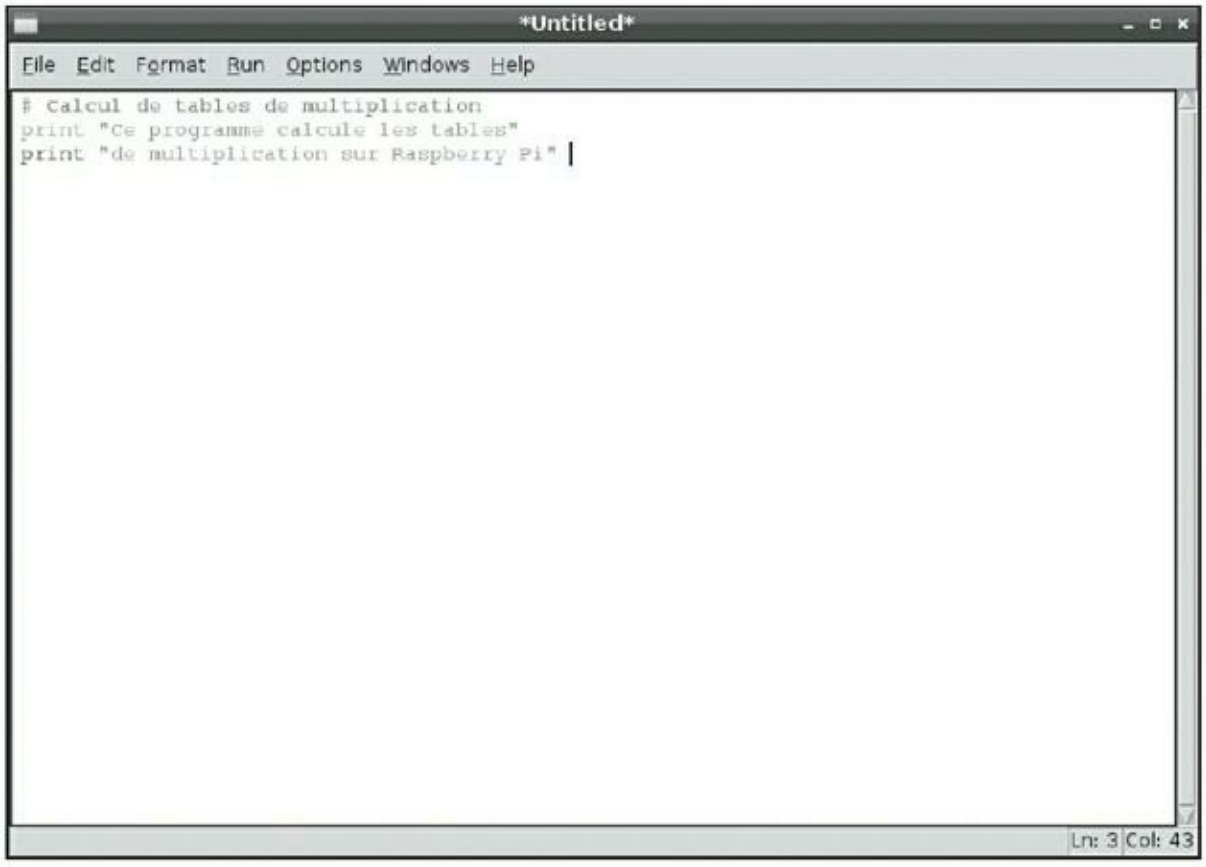

Untitled
File Edit Format Run Options Windows Help
Calcul de tables de multiplication
print "Ce programme calcule les tables"
print "de multiplication sur Raspberry Pi"
Ln: 3 Col: 43

Figure 11.2 : Séance d'édition Python.

Démarrer l'exécution d'un programme (le « lancer »), revient à demander à l'interpréteur de lire les instructions l'une après l'autre sans s'arrêter. Pour lancer le programme, ouvrez le menu **Run** et choisissez la commande **Run Current Script**. Vous prendrez vite l'habitude d'utiliser le raccourci clavier **F5**. Lorsque vous lancez le programme, Python doit afficher deux lignes dans le panneau de l'interpréteur.

Félicitations ! Vous venez d'écrire votre premier programme Python !

Vous devez enregistrer dans un fichier votre programme avant de pouvoir le lancer. Si vous ne l'avez pas encore enregistré, ou si vous avez effectué des modifications depuis les derniers enregistrements, Python vous demande d'enregistrer le programme lorsque vous tentez de le lancer. Cette action écrase l'ancienne version du programme dans le fichier. Dans le menu **File** de la fenêtre en mode script, il y a une option pour enregistrer sous un autre nom une copie du programme (**Save as**). Pensez à vous en servir pour créer plusieurs versions de votre programme en cours pour pouvoir revenir à une version antérieure lorsque vous ne comprenez plus ce qu'il se passe.

Utiliser des variables

La prochaine étape de notre projet consiste à demander à l'utilisateur de saisir le chiffre pour lequel nous allons lui générer une table de multiplication. Ce chiffre sera stocké en mémoire, dans une variable. Si vous avez fait les exercices Scratch, vous savez qu'une variable est une technique permettant de stocker une valeur (numérique ou texte) dans la mémoire en y associant un nom.

Imaginons que nous voulions stocker notre solde bancaire dans une variable. La valeur va parfois monter (super !) et parfois baisser (bien trop souvent). Dans tous les cas, ce sera votre solde bancaire. Précisons que les variables constituent l'un des fondements de la programmation, et pas seulement dans Python.

Si nous prenons l'exemple d'un solde bancaire, nous pouvons créer une variable en Python en lui donnant un nom approprié et en lui fournissant une valeur initiale, comme ceci :

```
solde = 500
```

Vous pouvez ensuite modifier la valeur (c'est pourquoi on appelle cela une variable) simplement en donnant cette nouvelle valeur :

```
solde = 250
```

Pour une telle variable, nous aurons surtout besoin de faire des additions et des soustractions, pour correspondre aux recettes et aux dépenses. Nous pouvons diminuer la valeur de la variable en lisant d'abord sa valeur actuelle puis en écrivant le résultat du calcul dans la même variable :

```
solde = solde – 250
```

Cette instruction lit la valeur actuelle de la variable `solde`, lui enlève 250 puis écrit le résultat dans la même variable. Pour afficher la valeur d'une variable à l'écran, vous utilisez la fonction `print()` en indiquant le nom de la variable :

```
print(solde)
```

Les programmeurs aiment utiliser une version abrégée des opérations arithmétiques appliquées aux variables. Pour une addition, la forme abrégée s'écrit `+=`, et pour la soustraction, elle s'écrit `-=`. Voici un exemple :

```
solde = 500
solde += 20
print(solde)
```

Si vous exécutez ce petit extrait, vous verrez s'afficher la valeur 520. L'exemple de décaissement suivant soustrait la valeur 70 de la valeur initiale pour afficher 430.

```
solde = 500
solde = -70
print(solde)
```

Cette écriture abrégée constitue une manière élégante de changer la valeur d'une variable. Vous la rencontrerez souvent dans Python.

Récupérer une saisie utilisateur

Avant d'aller plus loin, il est de notre devoir de clarifier un terme spécifique : celui de **fonction**. Une fonction est un sous-ensemble regroupant plusieurs commandes en vue de réaliser une tâche particulière. Python est livré avec une vaste panoplie de fonctions prédéfinies. Nous verrons plus loin comment définir vos propres fonctions dans ce même chapitre. Pour utiliser une fonction, il suffit d'indiquer son nom en le

faisant suivre d'un jeu de parenthèses. Lorsque la fonction attend une valeur en entrée pour faire varier son comportement, vous placez cette ou ses valeurs entre les deux parenthèses, comme nous l'avons fait pour `print()`.

Dans notre projet, nous voulons demander à l'utilisateur de saisir la base de sa table de multiplication pour pouvoir stocker la valeur dans une variable. Nous choisissons d'appeler cette variable `tablemul`. Nous allons directement stocker dans la variable `tablemul` la valeur renvoyée par un appel à la fonction standard `input()`. Cette fonction prédéfinie affiche un message à l'écran pour inviter l'utilisateur à saisir quelque chose, puis récupère ce qui est saisi et le renvoie.

Voici comment fonctionne cet appel à la fonction `input()` :

```
tablemul = input("Quelle table voulez-vous que je calcule ? ")
```

Vous remarquez que nous avons pris soin d'ajouter une espace après le point d'interrogation et avant le guillemet fermant la chaîne de caractères de la question. Le but est que le curseur de saisie ne soit pas collé au signe, ce qui est plus professionnel et facilite le repérage de l'endroit où apparaît ce qui est saisi.

Vous pouvez ajouter cette instruction à votre programme actuel et en lancer l'exécution. Vous devrez constater que le programme affiche la question puis présente un curseur en attendant que vous saisissiez un chiffre. Saisissez un chiffre et validez. Pour l'instant, le programme ne fait rien de votre chiffre, parce que nous ne lui avons pas encore expliqué quoi en faire.

Afficher des mots, des variables et des chiffres ensemble

Nous allons d'abord afficher un titre pour la table de multiplication que nous allons calculer. Cela nous amène à découvrir une nouvelle technique : l'affichage d'un texte combiné à l'affichage de la valeur d'une variable dans la même ligne. La fonction `print()` permet de réunir des choses différentes dans la même ligne, mais il faut les séparer par des virgules. Voici comment nous allons pouvoir combiner un texte et la valeur de notre variable `tablemul` :

```
print("\nVoici la table de multiplication des ",
tablemul, " :")
```

Les deux premiers caractères `\n` constituent un code spécial ou code *d'échappement* qui demande un passage à la ligne suivante. Cela nous permet de laisser un peu d'espace entre la question demandant la saisie et le titre du résultat.

N'oubliez pas que tout ce que vous indiquez entre guillemets sera affiché tel quel. Si vous citez le nom de la variable **tablemul** entre guillemets, vous verrez s'afficher le nom « tablemul » et non la valeur qui aura été lue dans la variable.

Il nous faut ensuite afficher une ligne pour chaque entrée dans notre table de multiplication, de 1 à 12. Vous savez déjà qu'il est possible d'utiliser des variables dans les calculs et d'imprimer le résultat. Voici comment nous pouvons construire notre table de multiplication :

```
print("1 fois ", tablemul, " font ", tablemul)
print("2 fois ", tablemul, " font ", tablemul*2)
print("3 fois ", tablemul, " font ", tablemul*3)
print("4 fois ", tablemul, " font ", tablemul*4)
# etc.
```

Si nous lançons l'exécution par **Run**, et demandons la table des 7, le résultat va nous surprendre :

```
1 fois 7 font 7
2 fois 7 font 77
3 fois 7 font 777
4 fois 7 font 7777
```

Ce n'est pas vraiment ce que nous voulions obtenir ! La variable `tablemul` est en fait considérée comme si elle contenait du texte, des lettres et des chiffres (une chaîne de caractères), donc rien qui puisse s'additionner comme des nombres. C'est le mode normal de la fonction de saisie `input()`. Quoi que l'utilisateur saisisse, le programme va afficher ce qu'il a récupéré, ce qui constitue une sorte de répétition de sécurité, comme dans les grands navires.

Pour résoudre ce souci, il suffit de demander de convertir la donnée saisie depuis son type initial vers le type dont nous avons besoin, ici le type numérique entier (*integer* en anglais). C'est le seul rôle de la fonction

nommée `int()`. Elle donne comme résultat un nombre sans partie décimale et accepte en entrée soit des chiffres, soit un nombre avec partie décimale (nombre à virgule flottante ou réel). Voici comment l'utiliser dans notre projet :

```
tablemul = int(tablemul)
```

Vous pouvez essayer le programme qui doit maintenant fonctionner comme dans la Figure 11.3. Pourtant, la solution n'est pas très satisfaisante. Il nous a fallu rédiger une ligne d'instruction pour chaque ligne de sortie, avec un calcul différent en fin de ligne. Bien sûr, nous pouvons profiter de la fonction copier/coller en mode script (dans le menu **Edit**), mais c'est une approche bête qui nous fait perdre patience. Comment ferions-nous si nous voulions créer une table allant jusqu'à 50 ou 500 ou 5000 ? Il nous faut une solution plus intelligente.

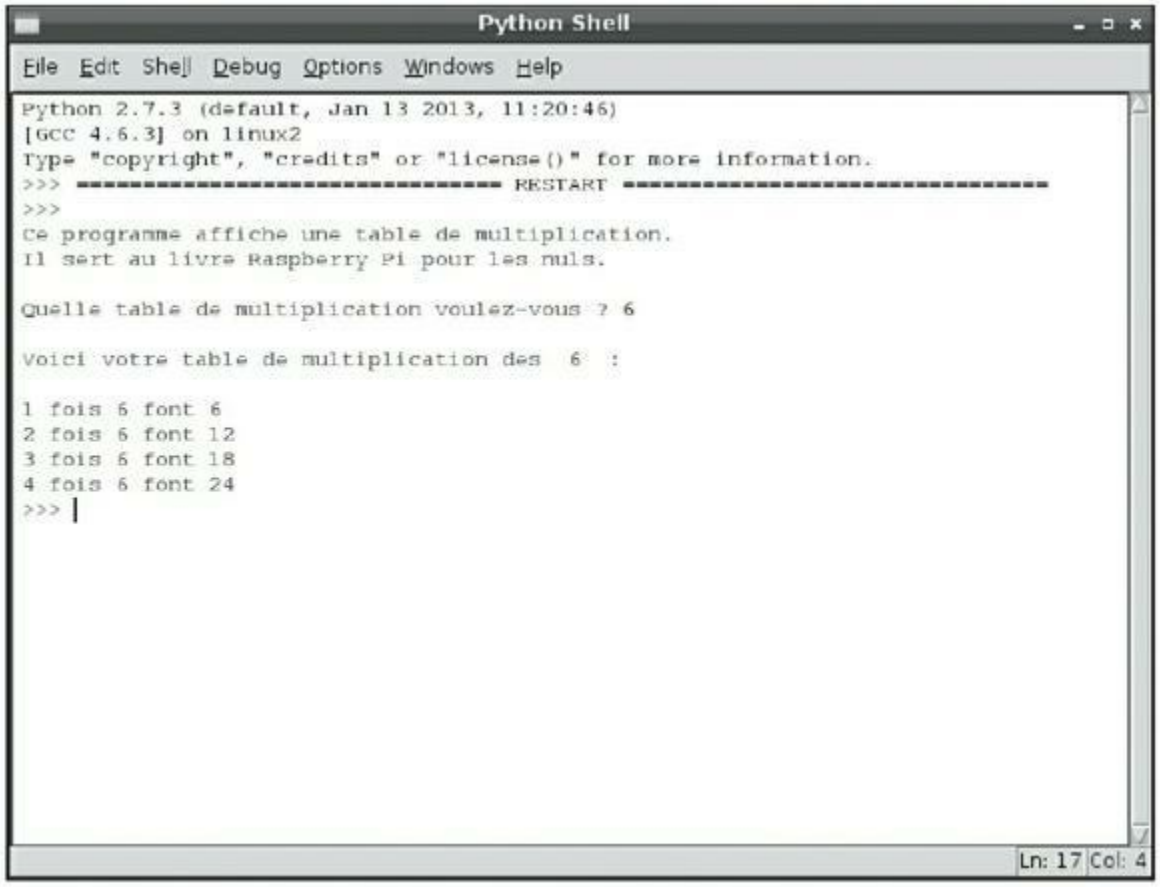

Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
Ce programme affiche une table de multiplication.
Il sert au livre Raspberry Pi pour les nuls.
Quelle table de multiplication voulez-vous ? 6
Voici votre table de multiplication des 6 :
1 fois 6 font 6
2 fois 6 font 12
3 fois 6 font 18
4 fois 6 font 24
>>>
Ln: 17 Col: 4

Figure 11.3 : La première version de la table de multiplication.

Répéter des instructions avec une boucle for

Pour nous épargner cette saisie d'instructions presque identiques à répétition, et pour rendre notre programme plus souple, nous allons adopter une boucle de répétition basée sur `for`. Toutes les instructions contrôlées par cette boucle seront répétées un certain nombre de fois. Dans chaque tour de boucle, nous allons augmenter la valeur d'une variable. C'est exactement ce dont nous avons besoin pour notre table de multiplication. Nous allons afficher une ligne pour chaque nombre entre 1 et 12 en affichant le résultat de la multiplication du nombre en cours par le nombre saisi.

Voici une boucle `for` qui permet de produire ce résultat :

```
for i in range(1,13):
    print(i, "fois", tablemul, "égal", i *
tablemul )
```

Ces deux lignes inoffensives contiennent pourtant plusieurs nouveaux concepts de programmation. Voyons d'abord la fonction nommée `range()`. Elle permet de créer une liste de nombres. Vous lui fournissez le premier nombre (1) et le dernier nombre (13), cette dernière valeur n'étant jamais touchée pour des raisons que nous vous expliquerons plus loin. Il nous faut donc indiquer 13 pour pouvoir multiplier jusqu'à 12.

La fonction `range()` peut être utilisée en dehors d'une boucle `for`, comme ceci (les trois chevrons montrent que nous sommes revenus au niveau de l'interpréteur) :

```
>>> print(range(5,15))
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

Vous pouvez spécifier une troisième valeur numérique entre les parenthèses de l'appel à la fonction pour indiquer la largeur du pas de progression d'un nombre au suivant. Nous n'en avons pas besoin ici, mais voici le principe :

```
>>> print(range(5,15,2))
[5, 7, 9, 11, 13]
```

Revenons à nos deux lignes d'instructions de début de section. La fonction `range()` génère une liste de nombres entre 1 et 12. Dans la même première ligne de la boucle, nous mentionnons une variable nommée `i` qui doit recevoir tour à tour chacune des valeurs produites par l'appel à la fonction `range()`. Au premier tour de boucle, `i` recevra la

valeur 1, et au second tour, la valeur 2. La valeur va ainsi jusqu'à la dernière répétition, au cours de laquelle `i` contiendra la valeur 12.

Un point extrêmement important décontenance les personnes ayant utilisé d'autres langages de programmation : dans Python, les lignes d'instructions qui font partie d'un bloc, par exemple celles d'une répétition `for`, doivent être indentées, par convention avec 4 espaces. Dans Python, ces espaces sont significatives, alors que dans la plupart des autres langages, vous pouvez placer autant d'espaces que vous voulez, puisque d'autres symboles servent à marquer le début et la fin d'un bloc. Cette obligation d'indenter les lignes rend automatiquement les programmes plus lisibles, car vous voyez immédiatement quelles sont les lignes qui dépendent de l'instruction de répétition `for`. Toutes les lignes dépendantes doivent bien sûr commencer à la même position depuis le bord gauche.

Nous pouvons donc faire répéter deux instructions en les indentant toutes deux :

```
for i in range(1,13):
    print(i, "fois", tablemul, "égal",
i*tablemul)
    print("------------------")
print("Cela vous convient ?")
```

Si votre boucle ne fonctionne pas, vérifiez que vous n'avez pas oublié d'ajouter le signe `:` à la fin de la première ligne du `for`.

Le précédent extrait progresse de 1 à 12 en affichant une ligne de table de multiplication à chaque fois puis une ligne de tirets en guise de séparateur. Une fois les douze lignes affichées, le programme affiche le message de conclusion. Vous remarquez qu'il n'est affiché qu'une fois parce que la ligne correspondante n'est pas indentée comme les précédentes.

Nous pouvons maintenant admirer le programme terminé :

```
# Exemple timestable_fr.py

print("Programme de calcul de table de
multiplication")
print("pour le livre Raspberry Pi pour les nuls")
```

```
tablemul = input("\nQuelle table voulez-vous que
je calcule ? ")

print("\nVoici la table de multiplication des "
tablemul, "" :")

for i in range(1,13):
    print(i, "fois", tablemul, "égal", i *
tablemul)
    print("-----------------" )

print("Cela vous convient ?")
```

Les indentations en début de lignes ont une signification spéciale, mais vous pouvez insérer autant de lignes vides que nécessaire pour augmenter la lisibilité du programme. C'est ce que j'ai fait dans l'exemple, en regroupant logiquement les différents sous-ensembles du programme. Vous constatez que nous avons aussi ajouté quelques codes de sauts de lignes `\n` dans les commandes d'affichage et demandes de saisie pour aérer l'affichage pendant l'exécution.

La plupart des gens apprennent plus efficacement en saisissant réellement le code source, mais vous pouvez vous contenter de télécharger le fichier du programme puis de l'ouvrir dans votre atelier pour gagner du temps.

La Figure 11.4 montre un exemple d'exécution de ce projet. Si vous voulez le personnaliser, vous pouvez par exemple faire calculer jusqu'à 20 ou n'afficher que les lignes impaires (1, 3 , 5). Dans les deux cas, il suffit d'intervenir sur les paramètres de la fonction `range()` dans la boucle. Vous pouvez aussi améliorer l'affichage en fournissant plus d'instructions ou bien en supprimant tous les messages inutiles. Vous pourriez même utiliser certains caractères tels que les tirets et les barres verticales pour constituer une sorte de cadre autour de votre table de multiplication.

Si le bouton d'exécution **Run** est grisé, vérifiez que vous n'avez pas laissé le programme tourner et arrêtez-le avec le bouton rouge **Stop**.

Si le résultat semble bizarre, ouvrez le menu **View** et choisissez **Variables** pour afficher à droite le panneau des variables. Vous avez ainsi accès à la valeur de chaque variable, ce qui est très utile pour mettre au point les programmes, d'autant plus lorsqu'ils deviennent complexes.

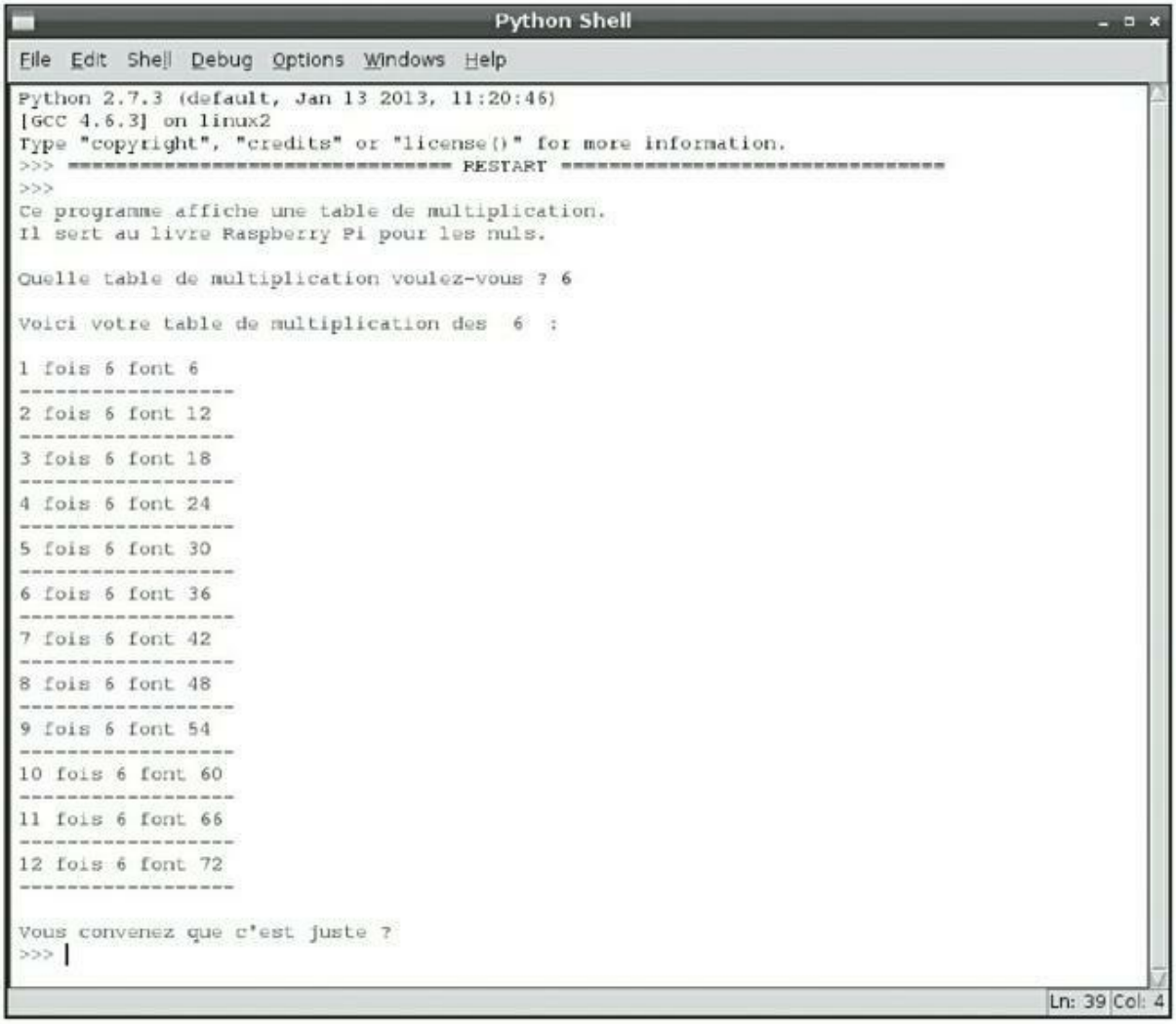

Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
Ce programme affiche une table de multiplication.
Il sert au livre Raspberry Pi pour les nuls.
Quelle table de multiplication voulez-vous ? 6
Voici votre table de multiplication des 6 :
1 fois 6 font 6
2 fois 6 font 12
3 fois 6 font 18
4 fois 6 font 24
5 fois 6 font 30
6 fois 6 font 36
7 fois 6 font 42
8 fois 6 font 48
9 fois 6 font 54
10 fois 6 font 60
11 fois 6 font 66
12 fois 6 font 72
Vous convenez que c'est juste ?
>>>
Ln: 39 Col: 4

Figure 11.4 : Le projet de table de multiplication terminé. Au fait, combien font 7 fois 8 ?

Création du projet Chatbot

Aimeriez-vous parler à votre ordinateur et le voir vous répondre intelligemment ? Notre second projet va vous donner l'occasion de discuter avec votre machine. Nous allons utiliser quelques astuces pour laisser croire que le programme a une certaine intelligence et qu'il tient compte de ce que vous lui indiquez. Bien sûr, ce n'est pas un vrai programme d'intelligence artificielle, car il s'agit d'une discipline informatique très complexe. Ici, c'est une simple démonstration. Le programme Chatbot (un robot pour chatter) pourra pourtant vous surprendre, et libre à vous d'augmenter son vocabulaire pour le rendre encore plus malin. Si vous voulez voir ce qu'il est capable de répondre, allez jeter un œil sur la Figure 11.5 en fin de chapitre.

La création de ce programme va vous permettre de renforcer vos connaissances du langage Python. Nous allons découvrir les instructions

conditionnelles, les listes, les dictionnaires et les choix aléatoires.

Voici les grandes lignes du fonctionnement de ce programme :

1. **Tout d'abord, il se présente puis invite le joueur à saisir une réponse**.
2. **Le joueur saisit quelques mots**.
3. **Si ce que le joueur a saisi contient le mot bye, l'ordinateur répond par un message de remerciement puis le programme se termine**.
4. **Le programme possède tout un stock de réponses**. Il va voir s'il reconnaît un des mots saisis par l'utilisateur en correspondance avec l'une des réponses. S'il en trouve une, il affiche cette réponse. S'il en trouve plusieurs, il en choisit une au hasard.
5. **Si aucun des mots n'est reconnu, le programme choisit une phrase au hasard**. Pour ne pas répéter la phrase affichée au hasard, il la remplace par celle saisie par le joueur. Le programme apprend donc de ce que le joueur saisit, et finit par parler comme lui.
6. **Le programme continue à discuter avec le joueur tant que ce dernier n'a pas saisi le mot-clé bye**.

Une fois que vous connaissez l'objectif, vous allez pouvoir faire vos premiers pas.

Si vous ne voulez pas le saisir, vous pouvez télécharger le programme source du fichier archive comme indiqué dans l'introduction.

Le concept de liste

Python offre plusieurs moyens pour stocker et manipuler des données. Les deux principales catégories de données en informatique sont les données simples et les données complexes. Un chiffre, un caractère sont des données simples. Une des techniques essentielles pour gérer des données complexes est la liste. Nous en avons déjà utilisé une avec la fonction `range()` pour créer une liste de nombre dans la boucle `for`. Vous pouvez bien sûr créer vos propres listes. Une donnée complexe est une donnée contenant plusieurs valeurs.

L'extrait suivant montre comment créer une liste portant le nom `listachats`. Vous pouvez la saisir sur la ligne de l'interpréteur ou créer un petit programme pour pouvoir plus facilement la modifier. Si vous choisissez de créer un programme, pensez à l'exécuter au moins une fois pour que la liste soit créée en mémoire.

```
listachats=["oeufs",
            "bacon",
            "tomates",
"pain",
"haricots verts",
"lait"]
```

Le principe est le même que pour créer une variable simple. Vous indiquez d'abord le nom de la liste puis le signe égal puis un crochet ouvrant (auquel répond un crochet fermant tout à la fin de la liste). Chaque élément de la liste est séparé du suivant par une virgule. Dans notre exemple, les éléments sont des mots, c'est-à-dire des chaînes de caractères, et c'est pourquoi nous devons ajouter des guillemets pour que Python sache où commence chaque élément. C'est notamment indispensable lorsque la chaîne comporte plusieurs mots séparés par une espace ou des virgules (comme dans « fromage, cheddar »). Sans ces délimiteurs que sont les guillemets, le programme pourrait croire qu'il y a plusieurs éléments.

Python accepte aussi bien les guillemets que les apostrophes pour délimiter les chaînes dans votre liste, mais nous vous conseillons d'utiliser des guillemets. En effet, les phrases des langages humains contiennent plus souvent des apostrophes, notamment en anglais et en français. Le principe est qu'il ne faut pas utiliser ce qui sert de délimiteur à l'intérieur de la chaîne. Si vous ajoutez une mention entre guillemets dans la chaîne délimitée par des guillemets, Python va croire que la chaîne se termine plus tôt que prévu. Pour utiliser ce qui sert de délimiteur dans la chaîne, vous devez le neutraliser en le faisant précéder par une barre oblique inverse, comme dans « J'ai dit \"Stop ! \" ».

En théorie, vous pouvez réunir tous les éléments de votre liste sur la même ligne, mais la lecture sera beaucoup plus facile en les aérant, une ligne par élément. Dans IDLE, lorsque vous frappez la touche Entrée à la fin d'un premier élément de liste, l'atelier devine que vous êtes dans une liste et indente automatiquement de la même largeur pour saisir l'élément suivant. Votre liste est donc proprement présentée comme dans mon exemple.

Lorsque vous saisissez une liste, méfiez-vous notamment des virgules. Il doit y en avoir une après chaque élément, sauf pour le dernier. C'est une autre raison pour répartir les éléments sur des lignes séparées, car vous verrez plus facilement s'il manque une virgule.

En guise d'exercice facultatif, vous pouvez faire directement afficher le contenu d'une liste à l'écran comme pour une variable simple. Voici en exemple :

```
>>> print(listachats)
['oeufs', 'bacon', 'tomates', 'pain', 'haricots
verts', 'lait']
```

Vous remarquerez que Python délimite les éléments de la liste par des apostrophes, même si vous aviez utilisé des guillemets comme délimiteurs. Cela n'a aucune importance. Pour connaître le nombre d'éléments dans une liste, vous profitez de la fonction `len()` :

```
>>> print(len(listachats)
6
```

Que faire si vous aviez oublié un élément ? Vous pouvez facilement en ajouter un à la fin de la liste au moyen de la fonction prédéfinie `append()`, comme ceci :

```
>>> print(listachats)
['oeufs', 'bacon', 'tomates', 'pain', 'haricots
verts', 'lait']

>>> listachats.append("poisson")

>>> print(listachats)
['oeufs', 'bacon', 'tomates', 'pain', 'haricots
verts', 'lait', 'poisson']
```

Python associe un numéro d'ordre à chaque élément de la liste en commençant à 0. Autrement dit, le deuxième élément porte le numéro 1, et le troisième le numéro 2. Ces numéros s'appellent des indices. Pour désigner un élément, vous pouvez indiquer son indice entre crochets droits, comme ceci :

```
>>> print(listachats[3])
pain
```

La ligne précédente affiche le contenu du quatrième élément, ne l'oubliez pas. Le premier élément possède l'indice 0. Vous pouvez même modifier un élément dans la liste au moyen de son indice. Voici par exemple comment changer le contenu du quatrième élément pour qu'il contienne baguette :

```
>>> listachats[3]="baguette"
```

Dans notre projet Chatbot, nous n'avons pas besoin d'en savoir plus au sujet des listes, mais sachez qu'elles représentent une technique très puissante pour structurer des données et les manipuler. Le Tableau 11.2 donne un petit aperçu de quelques fonctions applicables aux listes. Vous pouvez essayer ces exemples si vous êtes curieux.

Tableau 11.2 : Autres opérations sur les listes.

Action	Code	Notes
Tri d'une liste	`listachats.sort()`	Tri alphabétique ou dans l'ordre numérique croissant des indices.
Tri inverse	`listachats.sort(reverse=True)`	Tri dans l'ordre alphabétique inverse ou dans l'ordre des indices décroissants.
Suppression d'un élément	`del listachats[2]`	Suppression de l'élément désigné par l'indice. Les éléments suivants sont remontés d'un numéro pour ne pas laisser de trou.

Extraction d'un élément	`if "oeufs" in listachats : listachats. remove ("oeufs")`	Enlève l'élément mentionné, mais renvoie une erreur si l'élément n'existe pas. À utiliser dans une commande conditionnelle `if`.

Sachez pour vos projets futurs qu'une liste peut contenir des valeurs numériques aussi bien que des chaînes, et même une combinaison des deux. Voici par exemple une liste de réponse à un jeu de quizz :

```
repoquizz=["Isengrain", 1945, 2012,
"Suffragettes", 7500, "Albert Camus"]
```

La liste n'impose aucune règle quant à l'ordre des contenus, car Python n'a aucune idée de ce qu'elle contient, ni des règles de stockage. Pour que la liste prenne un sens, il faut écrire des instructions capables d'interpréter le contenu de la liste.

Créer une liste de réponses aléatoires

Puisque nous connaissons la structure, nous allons pouvoir créer un programme de réponses automatiques simplifié. Dans cette première version, nous allons demander au joueur de saisir un ou plusieurs mots, puis afficher une réponse au hasard et enfin remplacer la réponse par ce que le joueur a saisi.

Voici d'abord la totalité du code source du programme. Vous y repérerez des idées non encore présentées, mais nous verrons cela dans les descriptions qui suivent.

```
# Chatbot – version simple aléatoire (Chatbot1fr)
# Exemple du livre Raspberry Pi pour les nuls

import random

repaupif = ["Vraiment ?",
            "Tu en es certain ?",
            "Hmmmmm.",
```

```
            "Remarquable...",
            "Je ne suis pas certain
d'acquiescer...",
             "Absolument !",
             "Faut voir !",
            "Que disions-nous au fait ?",
            "Ce qui veut dire quoi ?",
            "Tu as sans doute raison.",
            "N'importe quoi ! Mais où on va, là
!",
            "Bref, tes projets pour demain ?",
            "Je me disais exactement la même
chose.",
            "Bien des gens sont de cet avis.",
            "Plusieurs personnes m'ont dit cela
déjà.",
             "Merveilleux !",
            "Cela pourrait s'avérer embarrassant
!",
            "Tu es vraiment de cet avis ?",
            "Effectivement...",
            "C'est ce que je voulais dire !",
            "Eventuellement..."]
print("A quoi penses-tu ?)
iladit = raw_input("Dis-moi tout : ")

repochoisie = random.randint(1, len(repaupif))-1
print(repaupif[repochoisie])
repaupif[repochoisie]=iladit
```

Les deux premières lignes sont les commentaires habituels qui expliquent en bref à quoi sert ce programme.

Dès l'origine, Python a été conçu pour être facile à enrichir, notamment en augmentant le nombre de fonctions prédéfinies. La troisième ligne, `import random`, indique que vous avez besoin d'utiliser une extension qui permet de générer des valeurs numériques aléatoires. Les extensions sont des modules, et nous en utiliserons plusieurs dans la création du jeu

vidéo du chapitre suivant, et dans les autres chapitres. Un module réunit un ensemble apparenté de fonctions déjà écrites et testées que vous pouvez directement réutiliser dans vos programmes, ce qui accélère et simplifie la programmation. Le module `random` regroupe une série de fonctions permettant de générer des valeurs numériques dans une séquence suffisamment imprévisible pour qu'elle puisse être considérée comme aléatoire. Nous en avons besoin pour sélectionner au hasard une des réponses prédéfinies que le programme doit afficher.

Dans la suite de notre code source, nous créons une liste nommée `repaupif` qui contient une liste de réponses que l'ordinateur va afficher en réaction à la saisie du joueur. Vous pouvez changer le texte des réponses ou en ajouter d'autres. Plus il y en aura, plus la machine semblera intelligente. Dans ce petit projet, quelques réponses suffisent à prouver que le programme fonctionne. L'ordre des réponses n'a aucune importance, mais méfiez-vous des virgules qui doivent être présentes à la fin de chaque ligne, sauf la dernière.

Après la liste, nous affichons un petit message pour inviter le joueur à partager ses pensées puis nous arrivons à l'instruction recueillant la saisie. Ici, nous n'avons pas utilisé la fonction `input()`, mais sa variante nommée `raw_input()`. Elle permet de saisir une chaîne de caractères et non une valeur numérique (elle n'existe plus en Python 3, mais dans Python 2.7 vous devez vous servir de cette variante pour les chaînes). Ce que le joueur va saisir au clavier est ensuite stocké dans une variable que nous nommons `iladit`.

L'instruction suivante est un peu plus complexe. Elle sert à obtenir au hasard un numéro d'indice. Pour mieux la comprendre, il faut la subdiviser. Voyons d'abord comment générer une valeur aléatoire. Nous appelons `random.randint()` en lui fournissant en argument deux valeurs entières (ce sont ses paramètres d'entrée). Les deux valeurs spécifient la plage de valeurs autorisées entre une borne basse et une borne haute. Pour afficher par exemple une valeur au hasard entre 1 et 10, vous écrivez ceci :

```
print(random.randint(1,10))
```

Vous pouvez tester la petite instruction précédente plusieurs fois pour voir qu'elle fonctionne bien. Il arrive que le même nombre revienne deux fois de suite, mais il en est ainsi avec les valeurs pseudo-aléatoires. Il vous arrivera la même chose lorsque vous lancez un dé. Même si c'est rare, il peut vous arriver de faire 6 fois de suite un 6.

La plage de valeurs autorisées est limitée par le nombre de chaînes dans la liste `repaupif`. Vous savez que nous pouvons utiliser la fonction `len()` pour connaître la taille de la liste. C'est ce qui nous permet d'ajouter ou d'enlever les éléments sans avoir à mettre à jour les indices à cet endroit du programme. Pour parer à tout agrandissement de la liste, nous remplaçons la seconde valeur par la valeur renvoyée par la fonction de calcul de la longueur de la liste :

```
print( random.randint(1, len(repaupif) ) )
```

Vous commencez sans doute à comprendre à quel point programmer peut devenir captivant.

Ici, nous ne nous contentons pas d'afficher l'indice. Nous devons le stocker dans une variable que nous nommons `repochoisie`. Il reste une dernière retouche à faire : vous savez que les indices des listes commencent à 0. Il nous faut donc ôter 1 à la valeur aléatoire que nous récupérons. Si nous oublions de prendre cette précaution, le programme ne pourra jamais choisir la première phrase, et pire encore, il tenterait parfois de lire une phrase plus loin que la fin de la liste (ce qui déclenche une erreur sévère). Voici donc l'instruction totalement écrite :

```
repochoisie = random.randint(1, len(repaupif)) -
1
```

Il ne reste que deux instructions pour afficher la phrase choisie au hasard puis recopier ce qui a été saisi à la place de la phrase venant d'être affichée :

```
print(repaupif[repochoisie])
repaupif[repochoisie]=iladit
```

Vous pouvez lancer le programme pour voir s'il fonctionne. Il nous manque encore une chose. Pour l'instant, vous n'avez le droit qu'à un essai. Pour que le programme ait envie de parler un peu plus longtemps avec vous, il nous faut découvrir le principe des boucles de répétition `while`.

La boucle de répétition while

Nous connaissons déjà la boucle fondée sur le mot-clé `for`, qui permet de répéter un groupe d'instructions un certain nombre de fois. Dans ce projet, nous voulons continuer à dialoguer avec le joueur tant qu'il n'a pas saisi le mot-clé `bye`. Dans ce cas, mieux vaut utiliser une boucle basée sur le mot-clé `while` (tant que).

La section que nous voulons répéter commence avec l'instruction qui demande au joueur de saisir quelque chose et se termine à la fin actuelle du programme, dans la ligne où nous insérons la saisie du joueur dans la liste de nos réponses automatiques.

Pour pouvoir répéter ce bloc, nous allons ajouter deux lignes au début puis décaler toutes les lignes existantes vers la droite en les indentant, afin que Python sache qu'elles font partie du bloc à répéter :

```
iladit=""
while iladit != "bye":
    iladit = raw_input("Dis-moi tout : ")
    repochoisie = random.randint(0,
len(repaupif))-1
    print(repaupif[repochoisie])
    repaupif[repochoisie] = iladit
```

Le principe de la commande `while` est de répéter l'exécution de toutes les instructions indentées tant que la condition du `while`, ce qui est à sa droite, reste vraie. Nous utilisons l'opérateur `! =` qui signifie « différent de ». La condition de notre `while` se lit ici `iladit ! ="bye"`. Cela signifie que l'expression renverra la valeur Vrai tant que le contenu de la variable `iladit` n'est pas égal à `bye`.

Dans la ligne de `while`, la variable `iladit` ne possède encore aucune valeur précise. Il faut donc l'initialiser pour ne pas déclencher une erreur. Voilà pourquoi nous avons ajouté une ligne juste avant l'instruction `while` pour créer la variable et lui donner la valeur vide, ce qui suffit à réussir le premier passage dans la ligne de `while` pour entrer dans la boucle. La valeur change presque tout de suite, dès que le joueur saisit quelque chose.

Si vous testez le programme maintenant, vous devriez constater que le dialogue se poursuit tant que vous n'avez pas tapé le mot `bye` dans votre saisie. Rappelons que vous pouvez augmenter l'intelligence du programme en augmentant le nombre de réponses dans la liste.

Ajout d’une sous-boucle pour forcer la saisie

Un autre besoin auquel nous pouvons répondre avec une boucle `while` va permettre d’empêcher le joueur de se contenter de frapper la touche Entrée sans rien saisir, soit volontairement, soit par mégarde. Nous évitons ainsi d’insérer dans notre superbe liste des chaînes vides. Dans un programme plus évolué, un tel contrôle de qualité de l’entrée de données est un point essentiel pour éviter les erreurs.

Vous pouvez tout à fait placer une boucle à l’intérieur d’une autre, ce qui s’appelle une *imbrication*. Nous allons donc mettre en place une petite boucle qui va sans cesse demander à l’utilisateur de saisir quelque chose de mieux que rien, et cette boucle sera dans la boucle principale de la conversation, celle qui se poursuit tant que l’utilisateur n’a pas saisi **bye**.

L’opérateur qui permet de tester si deux choses sont égales est constitué de deux signes égal consécutifs, ce qui peut dérouter les nouveaux programmeurs. Le signe égal isolé sert à affecter (à copier) une valeur, comme nous l’avons fait pour stocker une valeur dans une variable. Pour comparer deux choses, il faut redoubler le signe égal. Ce sont deux concepts totalement différents. Ne confondez pas l’opération de copie de ce qui est à droite du signe dans ce qui est à gauche avec l’opérateur de comparaison entre ce qui est à droite et à gauche du double signe.

Dans l’extrait ci-après, nous avons inséré notre sous-boucle `while` et une autre ligne. Dorénavant, tant qu’`iladit` est vide, le joueur doit saisir quelque chose. S’il se contente de frapper la touche Entrée, il n’a plus qu’à recommencer, et cela sans arrêt.

```
iladit = ""
while iladit != "bye":
    iladit = ""

    while iladit == ""
        iladit = raw_input("Dis-moi tout : ")

    repochoisie=random.randint(1,
len(repaupif))-1
    print(repaupif[repochoisie])
    repaupif[repochoisie]=iladit
```

Vous remarquez que nous avons indenté d'une deuxième série d'espaces la commande de saisie pour que Python sache que cette instruction fait partie de la sous-boucle de répétition (c'est la seule d'ailleurs).

Vous vous demandez certainement pourquoi nous avons ajouté un deuxième exemplaire de l'instruction pour vider le contenu de la variable pour la saisie du visiteur. La première fois, nous devions le faire parce que l'instruction `while` ne pouvait pas utiliser une variable qui n'existait pas encore. Dans ce deuxième cas d'utilisation, il nous faut vider ce que l'utilisateur a saisi, car au deuxième tour de boucle, `iladit` contient ce que le joueur avait saisi lors du premier tour. La sous-boucle ne fonctionnerait donc plus au deuxième tour puisque la variable testée ne serait pas vide. C'est un excellent exemple d'erreur de logique qui laisse apparemment le programme fonctionner. Il n'y aura pas de message d'erreur, ni de plantage, mais le résultat ne serait pas très intéressant, puisque le programme se contenterait de déblatérer tout seul sans vous laisser l'occasion de placer votre grain de sel.

Les dictionnaires de données { }

Nous savons utiliser la structure de données liste, mais il en existe une autre qui s'appelle le *dictionnaire*. Pour lire la valeur d'un élément d'une liste, nous utilisons un numéro d'indice qui correspond à la position de l'élément. Il en va différemment dans un dictionnaire : pour accéder à un élément, vous devez indiquer la clé qui est une chaîne ou un nombre identifiant l'élément. Ce principe est très utilisé en informatique. Votre numéro de compte bancaire, par exemple, est associé à vous seul ; il constitue donc une clé unique pour accéder à vos données. Un dictionnaire est plus évolué qu'une liste, car vous n'avez plus besoin de savoir quelle est la position de l'élément désiré dans la liste. Il suffit de connaître sa clé.

La création d'un dictionnaire se fonde sur des accolades (et non des crochets droits) qui délimitent des paires d'éléments, qui sont des couples constitués d'une clé et d'une valeur. Si cela vous paraît complexe, lisez directement un exemple. La manière dont les données sont structurées est très proche de celle d'un dictionnaire.

```
dicorepo = {"content":"Moi aussi, je suis content
!",
            "salut":"Bonjour à toi aussi !",
           "triste":"Cela ne peut que s'arranger
!",
```

```
            "raspberry":"J'adore les
framboises.",
            "ordinateur":"Nous allons prendre le
pouvoir !",
            "musique":"Le dernier quatuor
d'Olivier Greif ?",
            "art":"Est-ce de l'art ou du cochon
?",
            "lol":"Comment tuer un cirque ? Il
faut viser le jongleur.",
            "python":"Un bon tuyau sur les
serpents ?",
            "stupide":"C'est celui qui dit qui
l'est, dit-on ?",
            "temps":"Le soleil brillera d'ici le
réveillon.",
            "toi":"Je n'ai rien à voir dans cette
affaire !",
             "bof":"De quel film, la musique ?",
            "parle":"Blablabla. Agir fait du bien
aussi !",
            "pense":"Il faut panser ses
blessures.",
            "costume":"Je n'en ai pas besoin."}
```

Dans cet exemple, nous avons choisi le nom `dicorepo`, mais vous pouvez en choisir un autre. Vous devrez d'ailleurs donner des noms différents si vous utilisez plusieurs dictionnaires dans le même programme.

Dans ce dictionnaire, nous allons chercher un des mots de la phrase saisie pour trouver la réponse correspondante. Si le joueur tape le mot « content », l'ordinateur doit répondre « Moi aussi, je suis content. ». Avec le mot « salut », la réponse de l'ordinateur sera « Bonjour à toi aussi. ». Chaque entrée du dictionnaire se compose d'une clé et d'une valeur, avec un signe : (deux-points) entre les deux. Les différentes entrées sont séparées les unes des autres par une virgule.

Comme dans les listes, il faut faire très attention à la ponctuation dans les dictionnaires. Les chaînes sont délimitées par des guillemets, et le signe

deux-points (:) entre clé et valeur doit être à l'extérieur des guillemets. Chaque couple doit se terminer par une virgule sauf le dernier. L'ensemble des couples est délimité par une paire d'accolades (et non de crochets comme dans les listes).

Dans un dictionnaire, une même clé ne peut exister qu'en un exemplaire. Si vous aviez par exemple deux entrées avec une clé `content`, Python ne saurait pas laquelle des deux choisir.

Les dictionnaires ne fonctionnent que dans un sens : vous ne pouvez pas partir de la phrase pour retrouver la clé. C'est exactement la même chose que dans un vrai dictionnaire. Vous pensez que vous réussiriez à trouver un mot en cherchant d'après sa définition ? En revanche, trouver une définition à partir d'un mot est très simple, et heureusement.

Voici comment afficher une valeur lue dans le dictionnaire :

```
>>> print(dicorepo["salut"])
Bonjour à toi aussi !
>>> print(dicorepo["temps"])
Le soleil brillera d'ici le réveillon.
```

Si vous demandez une clé qui n'existe pas dans le dictionnaire, vous déclenchez une erreur. Vous verrez un peu plus loin dans ce chapitre lors de la création de la fonction de recherche dans le dictionnaire comment tester si une clé existe ou pas.

Dans le programme complet, nous avons ajouté quelques autres phrases, et c'est à ce niveau que vous pouvez marquer votre personnalité. Les mots que vous placez dans le dictionnaire et les réponses que vous prévoyez vont donner l'apparence d'une certaine intelligence au programme. Quand vous aurez fini de tester la version initiale, vous pourrez peaufiner les deux vocabulaires. Testez ensuite le programme en notant le type de mots que vous aimez saisir et le type de réponses que vous voulez voir s'afficher. Vous allez ainsi pouvoir construire un vocabulaire plus élaboré pour votre Chatbot.

Créer une nouvelle fonction

Une des choses intéressantes que l'on peut faire dans Python comme dans de nombreux autres langages consiste à créer (définir) une *fonction* pour réunir sous un même nom tout un groupe d'instructions. Une fonction est en mesure de recevoir une ou plusieurs données depuis l'endroit d'où elle

est *appelée* (ce sont ses *arguments*). Elle va travailler avec ces données puis renvoyer un résultat.

Dans notre programme Chatbot, il nous faut une fonction pour simplifier la recherche des mots saisis en tant que clés dans le dictionnaire.

Pour pouvoir utiliser une fonction (l'appeler), il faut d'abord la définir, ce qui se fait avec l'instruction `def`. Pour indiquer les instructions qui constitueront le corps de la fonction, nous procédons comme nous le savons déjà par indentations sous la ligne de l'instruction `def`. Voici un premier aperçu de l'allure générale d'une fonction et de son mode d'utilisation :

```
# Exemple de fonction
def scruterdico(message):
    print("Laissez-moi scruter le dictionnaire
pour ", message)
    return "salut"

scruterdico("test message")

resultat = scruterdico("test message2")
print("La réponse est :", resultat)
```

Nous reviendrons sur ce programme un peu plus loin, mais voyons d'abord ce qui va s'afficher lorsqu'on exécute cet extrait :

```
Laissez-moi scruter le dictionnaire pour test
message
Laissez-moi scruter le dictionnaire pour test
message2
La réponse est : salut
```

L'extrait est bref, mais il contient presque tout ce qu'il faut savoir au sujet des fonctions. Tout d'abord, nous déclarons et définissons notre fonction avec l'instruction suivante :

```
def scruterdico(message):
```

Notre fonction va porter le nom `scruterdico()` et nous indiquons qu'elle accepte lors de son démarrage une donnée qui aura été transmise au moment de l'appel, cette donnée étant stockée dans sa variable appelée

`message`. La ligne suivante affiche un texte puis le contenu de cette variable, donc la valeur de la donnée transmise lors de l'appel à la fonction. La ligne suivante contient une instruction `return` qui provoque la fin de l'exécution de la fonction combinée au renvoi du message fourni en paramètre, qui est ici « salut ».

Une fonction est un véritable sous-programme. La variable `message` est invisible du reste du programme, c'est une *variable locale*. Lorsque vous écrivez une fonction, vous spécifiez des instructions dans son corps et vous terminez ce corps par une instruction `return` pour rendre le contrôle à l'endroit d'où la fonction a été appelée, en renvoyant une valeur.

Même si une fonction est présente dans votre code source, elle peut ne jamais être exécutée. En effet, une fonction n'est exécutée que lorsque vous appelez cette fonction. La définition doit se trouver avant le premier appel pour que Python sache ce que vous tentez d'appeler. Voici un exemple d'appel de fonction :

```
scruterdico("test message")
```

Nous demandons d'exécuter la fonction `scruterdico()` en lui transmettant en argument la chaîne « test message ». Dès que la fonction démarre, cet argument est stocké dans la variable locale nommée `message` puis la fonction affiche son message. Le texte « hello » est renvoyé par la fonction, mais pour l'instant, nous ne récupérons pas cette valeur renvoyée.

Justement, l'extrait suivant montre comment récupérer la valeur qui est renvoyée par une fonction. Au lieu d'appeler la fonction, nous indiquons un nom de variable qui va recevoir le résultat de l'exécution, comme ceci :

```
resultat = scruterdico("test message2")
print("La réponse est :", resultat)
```

Lorsque la fonction renvoie la chaîne « hello », celle-ci est copiée dans la variable `resultat` et la ligne suivante peut afficher ce qui a été récupéré.

Cet exemple simplifié suffit à illustrer quelques-unes des raisons qui font du concept de fonction un élément essentiel dans la plupart des langages de programmation :

- Une fonction permet d'utiliser plusieurs fois un même groupe d'instructions dans un programme. Dans notre exemple, nous nous sommes servis de la même fonction pour afficher deux messages différents, simplement en envoyant un autre argument lors de chaque appel. Dans vos projets plus complexes, la possibilité de réutiliser des blocs d'instructions rendra vos programmes plus simples à lire, plus courts et donc plus rapides à écrire.
- Les fonctions simplifient l'étude des codes source parce qu'elles donnent un nom et une structure à un ensemble d'instructions. Dès que quelqu'un lit le nom `scruterdico()` dans notre programme source, il devine à quoi sert la fonction. Pour l'instant, nos projets sont très simples, mais lorsque vous travaillerez sur des projets de plus grande ampleur, vous constaterez que la lisibilité est un point très important.
- Les fonctions allègent la maintenance et la mise à jour des programmes. Vous trouvez plus facilement vos repères et vous localisez plus vite les endroits que vous avez besoin de retoucher. De plus, les modifications ne doivent être faites que dans un seul endroit. Si nous trouvons plus tard une meilleure manière de balayer le dictionnaire, il nous suffit de modifier la fonction, le reste du programme ne changeant pas.
- Les fonctions simplifient le prototypage, ce que nous avons fait ici. Nous avons pour l'instant écrit un programme expérimental qui reçoit un peu de

> texte et renvoie un autre texte. Notre vraie fonction de balayage du dictionnaire va faire la même chose, sauf que pour l'instant nous envoyons toujours le même message. Dans la version complète, nous allons envoyer un message différent selon ce que le joueur a saisi. Nous pouvons tout à fait construire le reste du programme autour de ce prototype pour le tester puis terminer notre fonction de balayage du dictionnaire plus tard.

Fonction de balayage du dictionnaire

Puisque nous savons maintenant créer une fonction, nous allons concevoir celle qui va lire le texte saisi par le joueur et chercher une clé en correspondance. Nous allons nous servir de ce que nous avons appris au sujet des dictionnaires et des fonctions, et y ajouter de nouvelles techniques concernant les boucles, les chaînes et les instructions conditionnelles pour prendre des décisions.

La fonction que vous allez découvrir tient en 12 lignes, mais elle est assez touffue. Elle commence par récupérer ce qui a été saisi puis elle compare chacun des mots de ce qui a été saisi aux clés du dictionnaire. Dans certains cas, le joueur aura saisi plusieurs mots qui se trouvent dans le dictionnaire. Si le joueur a par exemple dit « J'aime la musique pop. », les deux mots « aime » et « musique » sont des clés de notre dictionnaire. Dans ce cas, nous affichons une des deux réponses au hasard. Inversement, le joueur peut n'avoir saisi aucun des mots-clés. Notre fonction doit aussi tenir compte de cette possibilité.

Avant de passer à l'étude détaillée de la fonction, voici son code source complet afin que vous puissiez voir comment les différents éléments s'imbriquent les uns dans les autres :

```
def scruterdico(message):
    message = message.lower()
    motsjoueur = message.split()
    repokool = []
    for unmot in motsjoueur:
```

```
            if unmot in dicorepo:
                unerepo = dicorepo[unmot]
                repokool.append(unerepo)
        if repokool:
            repochoisie = random.randint(1,
len(repokool))-1
            return repokool[repochoisie]
         else:
            return ""
```

La première ligne nous est déjà connue de notre prototype. Lorsque nous appelons la fonction, nous lui envoyons le message qui correspond à ce qu'a saisi le joueur, et cette donnée est stockée dans la variable locale `message`.

Les deux instructions suivantes utilisent un nouvel outil : les méthodes sur chaînes. Ce sont des sortes de fonctions prédéfinies que vous pouvez relier au nom d'une chaîne (par un point) pour y appliquer un traitement. La méthode `lower()` convertit tous les caractères de la chaîne en minuscules. Nous en avons besoin parce que le joueur peut utiliser des lettres majuscules, ce qui empêchera de trouver les correspondances avec les clés qui sont toutes en minuscules. Rappelons que pour les programmes, « Salut » et « salut » ne sont pas la même chose. L'autre méthode, `split()` décompose une chaîne en autant de sous-chaînes que de mots. Autrement dit, la préparation de la chaîne saisie consiste à forcer tous les caractères en minuscules puis à créer une liste de mots séparés que nous stockons dans la variable `motsjoueur`.

Du fait qu'il peut y avoir plus d'une réponse, nous déclarons une variable de type liste vide pour stocker les réponses et nous lui donnons le nom `repokool`.

Nous pouvons maintenant passer à la boucle qui va balayer la liste des clés du dictionnaire en fonction de la liste des mots saisis. Nous avons déjà utilisé une boucle `for` dont la variable de boucle était un nombre choisi dans une série. Ici, nous devons progresser de mot en mot. Lors de chaque tour de boucle, la variable de travail `unmot` va contenir le prochain élément de la liste des mots saisis. Dans la prochaine ligne, nous découvrons une nouvelle technique : l'instruction conditionnelle qui commence ici par `if`. Une telle instruction permet d'exécuter une chose ou une autre en fonction d'une condition. Vous utiliserez ce genre d'instruction dans tous les programmes. Dans notre projet, cela nous

servira à tester la possibilité qu'il n'y ait aucune clé dans le dictionnaire, ce qui risquerait de provoquer l'arrêt du programme et l'affichage d'une erreur :

```
if unmot in dicorepo:
    unerepo = dicorepo[unmot]
    repokool.append(unerepo)
```

La variable `unmot` contient toujours un des mots saisis par le joueur. L'instruction conditionnelle `if` cherche ce mot dans le dictionnaire et n'exécute les deux instructions suivantes que si le mot est trouvé. Vous remarquez l'indentation qui permet de désigner le bloc d'instructions conditionnel, c'est-à-dire les actions qui dépendent de l'instruction `if`. Si le mot est trouvé dans le dictionnaire, le programme cherche la valeur correspondante et il l'ajoute à la liste `repokool` au moyen de la fonction standard `append()`.

Nous répétons ce processus pour tous les mots saisis. La ligne suivante n'est pas indentée par rapport au début de la boucle `for`, ce qui confirme que nous sommes sortis de cette boucle.

En sortie de boucle, nous testons si la liste `repokool` contient au moins une réponse de la façon suivante :

```
if repokool:
```

Cette manière d'écrire abrégée doit se comprendre comme « Si `repokool` contient quelque chose ». Les deux instructions qui sont indentées ne sont donc exécutées que si la liste `repokool` contient au moins une phrase de réponse, ce qui n'est possible que si au moins un des mots saisis a été trouvé dans le dictionnaire. Si tout cela est vrai, nous décidons de renvoyer un des éléments de la liste `repokool`, et nous le choisissons au hasard dans la liste, puis utilisons `return` pour renvoyer l'élément sélectionné et sortir de la fonction.

Nous arrivons ensuite à un nouveau mot-clé, `else` qui signifie *sinon*. C'est une branche facultative d'un bloc conditionnel `if`. (Vous remarquez qu'il n'y a pas de `else` dans le premier `if`). Si aucun des mots saisis par le joueur n'a été trouvé dans le dictionnaire, ce qui fait que `repokool` est vide, ce sont les instructions dépendantes de `else` qui sont exécutées. Ici, il n'y en a qu'une, et c'est l'envoi d'un message vide

au programme qui a appelé la fonction. Le `return` marque la sortie de la fonction.

Création du sous-programme principal

Notre première version de Chatbot ne fournissait que des réponses au hasard. Il nous faut maintenant revoir la boucle principale du dialogue pour que nous testions la présence des mots dans le dictionnaire afin de répondre intelligemment ou d'afficher une réponse aléatoire que nous remplaçons par ce qui a été saisi. Dans cette version finale, nous réalisons une synthèse de toutes les techniques présentées dans ce chapitre.

Voici ce que nous rédigeons juste après le bloc de la commande qui demande à l'utilisateur de saisir du texte :

```
reporaspi = scruterdico(iladit)
if reporaspi:
    print(reporaspi)
else:
    repochoisie = random.randint(1,
len(repaupif))-1
    print(repaupif[repochoisie])
    repaupif[repochoisie] = iladit
```

Dès la première ligne, nous procédons à un appel à notre fonction `scruterdico()` en lui transmettant le message saisi puis en récupérant la réponse dans la variable `reporaspi`.

La ligne suivante vérifie que `reporaspi` n'est pas vide. Si c'est le cas, nous affichons le contenu. Nous arrivons ensuite dans la branche `else` qui permet de traiter l'absence d'une réponse dans le dictionnaire. Dans ce cas, nous choisissons une réponse au hasard, nous l'affichons puis nous la remplaçons par ce que le joueur a saisi.

Test de Chatbot

Notre programme Chatbot est terminé (le listing complet de la version française est fourni en fin de chapitre). Nous avons découvert les variables, les listes, les boucles de répétition, les valeurs aléatoires, les

dictionnaires, les instructions conditionnelles `if` et `else` et les fonctions. Nous savons comment récupérer une donnée saisie par l'utilisateur et comment afficher une réponse à l'écran. Nous avons conçu le squelette d'un programme de dialogue informatique que vous pouvez personnaliser *ad libitum.*

La Figure 11.5 montre un exemple de session. Vous remarquerez peut-être que l'ordinateur ne comprend pas bien ce que vous saisissez, mais il suffit d'augmenter le nombre de phrases pour qu'il réponde de mieux en mieux. Vous finirez par constater en enrichissant le dictionnaire que le programme peut même vous étonner et semble presque devenir intelligent. Vous ne serez plus jamais seul avec votre Raspberry Pi !

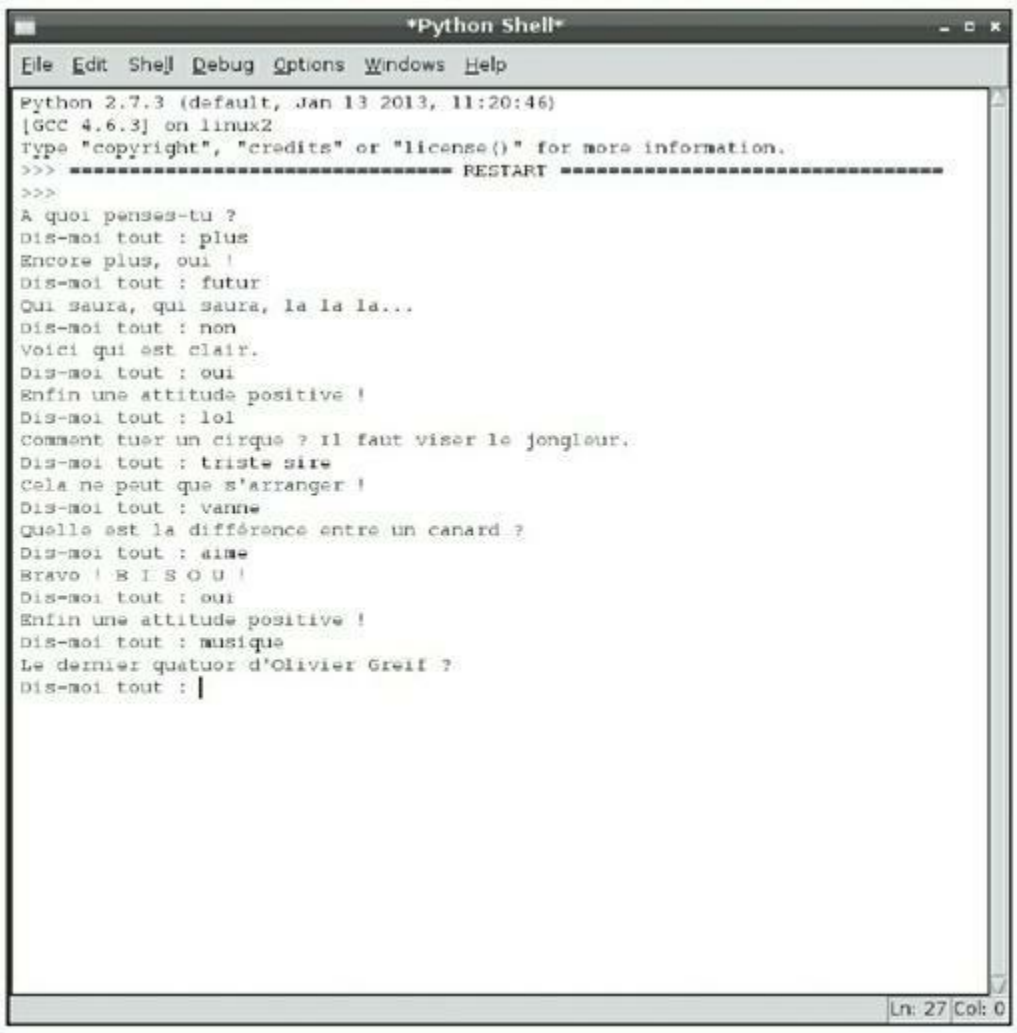

Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
A quoi penses-tu ?
Dis-moi tout : plus
Encore plus, oui !
Dis-moi tout : futur
Qui saura, qui saura, la la la...
Dis-moi tout : non
Voici qui est clair.
Dis-moi tout : oui
Enfin une attitude positive !
Dis-moi tout : lol
Comment tuer un cirque ? Il faut viser le jongleur.
Dis-moi tout : triste sire
Cela ne peut que s'arranger !
Dis-moi tout : vanne
Quelle est la différence entre un canard ?
Dis-moi tout : aime
Bravo ! B I S O U !
Dis-moi tout : oui
Enfin une attitude positive !
Dis-moi tout : musique
Le dernier quatuor d'Olivier Greif ?
Dis-moi tout :
Ln: 27 Col: 0

Figure 11.5 : Petite conversation avec notre Chatbot.

Code source complet de Chatbot

Pour plus de confort, voici le listing complet du programme Chatbot.

```
# chatbot – (chatbot_fr.py)
```

```
# Exemple du livre Raspberry Pi pour les nuls
# de Sean McManus et Mike Cook
# VF par O. Engler

import random

repaupif = ["Vraiment ?",
            "Tu en es certain ?",
            "Hmmmmm.",
            "Remarquable...",
            "Je ne suis pas certain
d'acquiescer...",
             "Absolument !",
             "Faut voir !",
            "Que disions-nous au fait ?",
            "Ce qui veut dire quoi ?",
            "Tu as sans doute raison.",
            "N'importe quoi ! Mais où on va, là
!",
            "Bref, tes projets pour demain ?",
            "Je me disais exactement la même
chose.",
            "Bien des gens sont de cet avis.",
            "Plusieurs personnes m'ont dit cela
déjà.",
             "Merveilleux !",
            "Cela pourrait s'avérer embarrassant
!",
            "Tu es vraiment de cet avis ?",
            "Effectivement...",
            "C'est ce que je voulais dire !",
            "Eventuellement..."]

dicorepo = {"content":"Moi aussi, je suis content
!",
             "salut":"Bonjour à toi aussi !",
```

```
            "triste":"Cela ne peut que s'arranger
!",
            "raspberry":"J'adore les
framboises.",
            "ordinateur":"Nous allons prendre le
pouvoir !",
            "musique":"Le dernier quatuor
d'Olivier Greif ?",
            "art":"Est-ce de l'art ou du cochon
?",
            "lol":"Comment tuer un cirque ? Il
faut viser le jongleur.",
            "python":"Un bon tuyau sur les
serpents ?",
            "stupide":"C'est celui qui dit qui
l'est, dit-on.",
            "temps":"Le soleil brillera d'ici le
réveillon.",
            "toi":"Je n'ai rien à voir dans cette
affaire !",
             "bof":"De quel film, la musique ?",
            "parle":"Blablabla. Agir fait du bien
aussi !",
            "pense":"Il faut panser ses
blessures.",
          "costume":"Je n'en ai pas besoin.",
            "vanne":"Quelle est la différence
entre un canard ?",
            "vannes":"Je n'en connais qu'une",
             "plus":"Encore plus, oui !",
            "internet":"87% des statistiques web
sont fausses, y compris
            celle-ci.",
             "aime":"Bravo  !  B I S O U  !",
            "haine":"Pas folichon. Soyons
positifs.",
```

```
            "futur":"Qui saura, qui saura, la la
la...",
            "demain":"J'arrête les frites.",
            "non":"Voici qui est clair.",
             "oui":"Enfin une attitude positive
!",
            "espoir":"Williams ou Conférence ?",
            "rêve":"Je crois avoir rêvé, mais ce
devait être en rêve.",
            "dire":"Je peux tout te dire à toi.",
            "you":"On ne parle pas anglais ici
!",
            "ok":"Je n'ai jamais le hoquet,
moi.",
             "es":"Qui, toi ?",
            "rire":"On peut toujours, mais aux
dépens de qui ?",
            "merde":"Pas question ! Ce mot est
trop gros pour ma mémoire."}

def scruterdico(message):
    message = message.lower()
    motsjoueur = message.split()
    repokool = []
    for unmot in motsjoueur:
        if unmot in dicorepo:
            unerepo = dicorepo[unmot]
            repokool.append(unerepo)
    if repokool:
        repochoisie = random.randint(1,
len(repokool))-1
        return repokool[repochoisie]
    return ""

print("A quoi penses-tu ?")
```

```
iladit = ""

while iladit != "bye":

    iladit=""
    while iladit == "":
        iladit = raw_input("Dis-moi tout : ")

    reporaspi = scruterdico(iladit)
    if reporaspi:
        print(reporaspi)
     else:
        repochoisie = random.randint(1,
len(repaupif))-1
      print(repaupif[repochoisie])
        repaupif[repochoisie] = iladit

print("Au revoir. Merci pour cette petite
discussion. A plus !")
```

Chapitre 12

Un jeu avec Python et Pygame Zero

DANS CE CHAPITRE :

- **Afficher des images et jouer des sons avec Pygame Zero**
- **Créer un jeu de clique-nuage**
- **Gérer plusieurs images avec une liste**
- **Animer des images**
- **Ajouter un compte à rebours**

Concevoir puis réaliser un jeu vidéo est une excellente manière d'apprendre à programmer. Cela vous permet d'obtenir rapidement des résultats à l'écran, afin de voir comment marche votre projet, tout en prenant du bon temps une fois le programme mis au point. Et la programmation de jeux vidéo suscite rapidement de nouvelles idées, et donc de nouveaux défis de programmation à relever pour améliorer le programme.

Dans la programmation en langage Python, un outil très utilisé pour créer des jeux vidéo porte le nom *Pygame*. Il s'agit d'une librairie de fonctions qui simplifie la gestion des images et des sons, parmi d'autres opérations. Cela dit, Pygame comporte certaines parties un peu complexes, et les premiers pas ne sont pas toujours faciles.

C'est pour résoudre cette relative difficulté d'accès que Daniel Poppe a créé la librairie nommée *Pygame Zero*, qui simplifie Pygame pour vous permettre de débuter plus facilement. Cette librairie comporte elle aussi de nombreuses fonctions pour gérer les animations, les boucles, les sons et les images. Au départ, Pygame Zero a été conçu dans un but éducatif, mais la librairie est tout à fait intéressante, même pour un non-débutant.

Pygame Zero fonctionne non seulement sous Linux, mais également pour les Mac et sous Windows. Les jeux que vous avez créés sur votre Raspberry Pi peuvent donc être utilisés par vos amis ou par vous-même sur d'autres machines.

N.d.T. : Le nom Pygame Zero n'a aucun rapport avec la carte Raspberry Pi Zero.

Nous allons voir dans ce chapitre comment créer le jeu qui va porter le nom *Cloudbusting* (chasseur de nuages). Ce jeu consiste à faire éclater en moins de 10 secondes le plus possible de nuages qui vont défiler du bas vers le haut de l'écran. Le jeu peut fonctionner sur une carte Raspberry ancienne comme la Modèle B+, mais l'exécution sera meilleure sur une Raspberry Pi 2 ou Pi 3.

Pour réaliser les exemples du chapitre, il vous faut quelques fichiers de sons et d'images, que je décris dans la liste qui suit. Vous pouvez télécharger les fichiers d'exemple (j'ai présenté dans l'introduction comment récupérer ces fichiers). Vous pouvez aussi créer vous-même des fichiers compatibles. Le jeu fonctionne quel que soit le contenu exact des images, à partir du moment où leur format est correct.

Récupérer des images et des sons

Voici les ressources qu'il vous faut pour le jeu *Cloudbusting* :

» Six images portant les noms *target0.png* jusqu'à *target5.png*. Dans notre exemple, nous avons choisi des nuages colorés. Chacune des images mesure environ 100 pixels sur 90 pixels, à 10 ou 20 pixels près dans les deux sens. (Nous avons remarqué que le fait de choisir une taille légèrement différente pour certaines images augmente l'effet de profondeur à l'écran. N'hésitez pas à essayer cette astuce.) Pensez à prévoir un fond transparent si vous créez vos images pour que le rectangle de l'image soit invisible, et ne vienne pas masquer les images qui apparaissent derrière. Le format graphique recommandé est PNG, mais la

librairie sait également utiliser les formats GIF et JPEG, sauf que ce dernier format ne permet pas la transparence. Si vous voulez créer vos images, une bonne solution consiste à utiliser l'éditeur de dessin de Scratch (revoyez le Chapitre 10). Une fois qu'une image est créée dans Scratch, il suffit de cliquer droit dans le costume de la zone des costumes puis de choisir la commande Enregistrement dans le menu local. Dans Scratch 1.4, l'opération est une exportation, et dans Scratch 2.0, c'est un enregistrement vers fichier local.

» Une image portant le nom *pop.png*. C'est cette image qui sera affichée dans chaque nuage quand vous l'aurez fait éclater par un clic. Dans l'exemple, nous avons choisi un phylactère avec la mention **POP !**. Nous avons créé l'image avec Scratch.

» Deux fichiers son, un pour le bruit de l'éclatement d'un nuage et l'autre comme signal sonore de fin de partie. Pour l'éclatement de nuage, nous sommes partis d'un effet sonore gratuit trouvé sur le Web et portant le nom *blop.ogg*. Pour la mélodie de fin de partie, c'est l'un des auteurs, Sean, qui l'a créée sur sa tablette et lui a donné le nom *whoops.ogg*.

Les fichiers sonores sont au format .ogg très répandu dans le monde du logiciel libre. La librairie Pygame Zero reconnaît aussi le format .wav. Un fichier doit porter le nom *blop.ogg* et l'autre le nom *whoops.ogg*. Si vous choisissez d'autres noms, il faudra modifier le code source en conséquence.

Vous pouvez bien sûr télécharger le code source complet de l'exemple, si vous préférez ne pas saisir les lignes vous-même. Je rappelle que j'ai

indiqué comment récupérer les fichiers dans l'introduction. Le texte source du programme complet est fourni à la fin du chapitre.

Préparer les dossiers

La librairie Pygame Zero est très stricte au niveau des noms que doivent porter les fichiers et des sous-répertoires dans lesquels il faut les stocker. Les noms des fichiers doivent toujours être en lettres minuscules, ce qui garantit une utilisation sans problème, quel que soit le système d'exploitation (Linux, Windows, macOS). Les images doivent être stockées dans un sous-dossier portant le nom *images* et les sons dans un sous-dossier *sounds*.

Préparons donc ces deux sous-dossiers comme ceci :

1. **Dans l'interface graphique, démarrez le Gestionnaire de fichiers (barre des tâches ou menus)**. J'ai présenté la barre des tâches et le Gestionnaire de fichiers dans le Chapitre 4.

N. d. T. : Notez qu'en français le Gestionnaire de fichiers porte un nom inutilement long, mais cet inconvénient sera sans doute corrigé quand vous lirez ces lignes.

2. **Placez-vous dans votre dossier principal *pi* puis cliquez droit pour choisir dans le menu local la commande Créer nouveau/Dossier.**
3. **Donnez au nouveau dossier le nom *images*.**
4. **Répétez les Étapes 2 et 3 en choisissant le nom *sounds*.**
5. **Copiez vos fichiers image dans le sous-dossier *images* et vos fichiers audio dans le sous-dossier *sounds*.**

Si vous avez téléchargé les fichiers depuis le Web, ils sont normalement stockés dans le dossier des téléchargements ou *Downloads* qui est dans votre dossier principal *pi*. Si le fichier est au format compressé ZIP, il faut

le décompresser en double-cliquant dans le nom du fichier puis en choisissant la commande d'extraction. Une fois les fichiers extraits, vous les répartissez dans les deux sous-dossiers *images* et *sounds*. Le Chapitre 4 a présenté les opérations de copie et de déplacement de fichiers.

Écrire et exécuter un premier programme

Les lignes de code du Listing 12.1 représentent votre premier programme Pygame Zero. Il se contente d'afficher une image au centre de la fenêtre. Saisissez ces lignes dans l'éditeur de Thonny ou dans celui de Python 3 IDLE puis enregistrez ce que vous avez saisi dans votre dossier *pi* sous le nom *nebu1.py*. J'ai présenté les environnements de développement Thonny et IDLE dans le Chapitre 11.

N. d. T. : Pour mieux comprendre les exemples, j'ai francisé plusieurs noms de variables et d'objets. Ainsi, tous les éléments qui étaient nommés **cloudx** sont devenus des **nebux** (de nébulosité). Plusieurs fonctions que nous allons définir doivent garder leur nom anglais car c'est ainsi qu'elles sont connues par le moteur du jeu Pygame Zero.

Listing 12.1 : Code source du premier programme Pygame Zero (nebu1.py).

```
WIDTH  = 500
HEIGHT = 500

nebu1 = Actor('target5')
nebu1.x = 250
nebu1.y = 250

def draw():
    screen.clear()
    nebu1.draw()
```

Exécution d'un programme Pygame

Zero

Pour lancer l'exécution d'un programme conçu avec Pygame Zero, vous n'utilisez pas une commande dans l'atelier. Vous devez écrire une ligne de commande. Voici comment faire :

1. **Depuis le Gestionnaire de fichiers, naviguez jusqu'à votre dossier principal *pi*.**

   Vous devez voir le fichier que vous venez de créer (*nebu1.py*) ainsi que les noms des deux sous-dossiers pour les images et les sons.

2. **Dans le menu principal du Gestionnaire de fichiers, choisissez Outils puis la commande Ouvrir le dossier actuel dans un terminal**.

   Vous pouvez aussi utiliser le raccourci clavier F4 pour obtenir le même résultat.

3. **Dans la fenêtre de terminal, saisissez la commande suivante puis validez par la touche Entrée :**

```
pgzrun nebu1.py
```

La Figure 12.1 montre le résultat attendu. Il y a une image fixe au milieu d'une nouvelle fenêtre. Il se peut qu'il faille attendre un peu l'ouverture de la fenêtre. Une fois que vous avez bien admiré ce résultat, vous refermez la fenêtre avec le bouton du coin supérieur droit ou bien au moyen du raccourci clavier **Ctrl + C** dans la ligne de commande. En revanche, ne refermez par la fenêtre du terminal, car cela vous évite de devoir la rouvrir à chaque nouvel essai. C'est de cette fenêtre que vous lancez l'exécution du même programme modifié ou d'un autre.

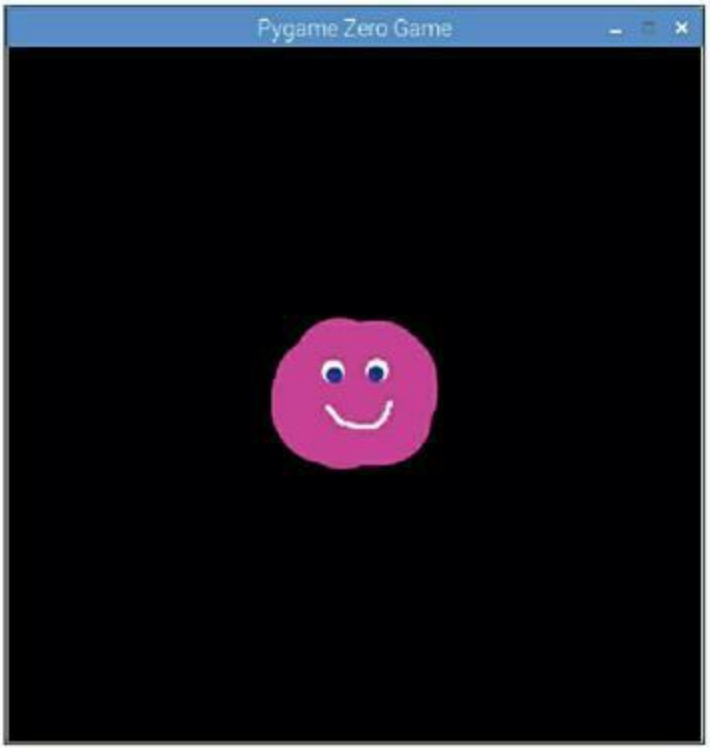


Figure 12.1 : Affichage du premier programme nebu1.

En étudiant le code source, vous comprenez qu'il est très facile de gérer des images avec Pygame Zero.

Les deux variables portant les noms WIDTH et HEIGHT servent à définir la taille initiale de la fenêtre. Dans notre exemple, c'est une fenêtre carrée de 500 pixels de côté.

N.d.T. : Bien qu'en théorie les deux variables WIDTH et HEIGHT auraient pu être traduites par exemple en LARGEUR et HAUTEUR, mes essais ont montré que cela avait un effet secondaire imprévu sur la largeur et la hauteur de la fenêtre. Je les ai donc laissées telles quelles.

Pour contrôler les éléments visuels mobiles, Pygame Zero parle d'acteurs, dans le même esprit que les lutins de Scratch (les *sprites*). À chaque acteur sont associées une image et des coordonnées, ce qui simplifie la gestion des personnages et des obstacles dans les jeux. Ce genre d'aide à la programmation existe également dans Scratch, mais pas dans le langage Python, ni même dans la librairie de jeu Pygame. Grâce aux acteurs de Pygame Zero, le passage de Scratch vers le langage Python en programmation est beaucoup plus simple.

Pour créer un acteur, il faut lui donner un nom et lui indiquer le nom du fichier d'image qui va incarner cet acteur à l'écran :

```
nebu1 = Actor("target5")
```

La ligne précédente déclare un acteur qui porte le nom **nebu1** et qui va être associé à une image se trouvant dans le fichier *target5.png*. Vous constatez qu'il est inutile de donner l'extension de nom de fichier, car Pygame Zero l'ajoute d'office (ce qui vous fait un souci de moins).

Les coordonnées ont leur origine dans le coin supérieur gauche de la fenêtre. Dans notre exemple, elles vont donc de 0 à 500 de gauche à droite et de haut en bas. Les coordonnées correspondent aux deux variables **x** et **y**. Pour faire afficher le nuage bien au centre dans le sens horizontal, nous pouvons définir sa coordonnée en X de la façon suivante :

```
nebu1.x = 250
```

Alors que les coordonnées de la fenêtre commencent en haut à gauche, les coordonnées des objets que sont les acteurs correspondent au centre de ces objets. En donnant à la variable **x** de l'acteur la valeur 250, nous faisons apparaître l'image à mi-chemin dans le sens horizontal.

Il ne reste plus qu'à faire afficher l'acteur. C'est l'objectif d'une fonction prédéfinie qui est exécutée périodiquement par la librairie Pygame Zero. (J'ai présenté le fonctionnement des fonctions dans le Chapitre 11.) Une fois que vous avez affiché un objet à l'écran, il reste affiché indéfiniment. Voilà pourquoi il faut d'abord effacer l'écran avant d'afficher de nouveaux objets :

```
screen.clear()
```

Dans notre exemple, nous nous contentons d'afficher ensuite le seul acteur à la position que nous lui avons définie, en écrivant l'appel à la fonction de dessin :

```
nebu1.draw()
```

Sauriez-vous ajouter les lignes pour afficher un deuxième nuage à gauche du premier ? Sauvegardez le projet sous le nom *nebu2.py* et ajoutez l'instruction pour créer le second nuage en adoptant par exemple le nom **nebu2**. N'oubliez pas de changer la valeur de la variable **x** du nouveau nuage et de demander son affichage. Le Listing 12.2 donne une solution possible. Vous pouvez choisir une autre valeur pour la position en X.

Listing 12.2 : Ajout d'un second nuage (nebu2.py).

```
WIDTH  = 500
HEIGHT = 500
```

```
nebu1 = Actor('target5')
nebu1.x = 250
nebu1.y = 250

nebu2 = Actor('target2')
nebu2.x = 80
nebu2.y = 250

def draw():
    screen.clear()
    nebu1.draw()
    nebu2.draw()
```

Lancez l'exécution depuis la ligne de commande, comme vous savez le faire maintenant :

```
pgzrun nebu2.py
```

Détecter les clics de souris

Pygame Zero vous permet facilement de faire réagir votre programme aux clics de souris. Il suffit d'ajouter par exemple la définition de fonction suivante à la fin du programme. Sauvegardez la version actuelle du programme sous le nom *nebu3.py* puis ajoutez les quatre lignes suivantes tout à la fin, en commençant bien sur la marge gauche pour la première ligne, après quatre espaces pour la deuxième ligne et après huit espaces pour les deux dernières :

```
def on_mouse_down(pos):
    if nebu1.collidepoint(pos):
        nebu1.image = 'pop'
        sounds.blop.play()
```

Cette définition de fonction utilise un nom de fonction prédéfini, **`collidepoint()`**. Elle va être appelée automatiquement lorsque vous cliquerez avec la souris. Les coordonnées du pointeur de souris au moment du clic vont être stockées dans la variable **pos**, qui est une variable composite, plus exactement un *tuple*.

Il n'est pas nécessaire de savoir ce qu'est un tuple pour l'utiliser dans Pygame Zero, mais vous serez sans doute heureux de savoir qu'un tuple ressemble un peu à une liste, sauf que vous ne pouvez pas modifier les valeurs qu'elle contient. Pour définir un tuple, vous utilisez des parenthèses et non des crochets droits. Le tuple portant le nom **pos** contient les deux variables **x** et **y** qui représentent les coordonnées du pointeur de souris au moment du clic.

Puisque nous connaissons les coordonnées de la souris, nous allons pouvoir lancer un test pour savoir si le clic a eu lieu dans les limites de l'acteur. Cela correspond à la technique de détection de collision qui consiste à comparer deux jeux de coordonnées, ici pour savoir si la surface de l'acteur contient les coordonnées du pointeur. Normalement, le code est assez difficile à écrire, mais avec Pygame Zero, tout est beaucoup plus simple. Il suffit d'utiliser une instruction conditionnelle basée sur **if** avec la fonction prédéfinie nommée **`collidepoint()`**. Si le test est positif, les deux lignes qui sont en retrait par rapport à celle du test **if** sont exécutées. La première provoque le remplacement de l'image actuelle de nuage par celle suggérant un son d'éclatement, *pop.png* :

```
nebu1.image = 'pop'
```

Ici non plus, nous n'avons pas besoin d'indiquer l'extension du nom de fichier image.

La deuxième ligne qui est exécutée si le test a réussi permet de faire entendre le son contenu dans le fichier *blop.ogg*. Ici non plus, nous ne fournissons pas l'extension .ogg, car Pygame Zero l'ajoute d'office. Ces quatre lignes montrent la technique de réaction à un événement, et vous pourrez vous servir de ces lignes comme modèle pour d'autres projets.

N. d. T. : Dans les deuxième et troisième lignes de cet extrait, le nom **nebu1** est un nom d'*objet*. Le nom **`collidepoint()`** est le nom d'une *méthode* de cet objet. La variable **image** de l'objet **nebu1** s'appelle une *donnée membre*. Ce sont les termes utilisés en programmation objet.

Vous avez sauvegardé cette version du projet sous le nom *nebu3.py* dans votre dossier *pi*. Vous pouvez donc lancer l'exécution pour voir le résultat avec la commande suivante dans la fenêtre du terminal :

```
pgzrun nebu3.py
```

Lorsque vous cliquez en dehors des nuages, il ne doit rien se passer. En revanche, si vous réussissez à cliquer dans le nuage du centre, vous devez voir apparaître l'image pop !, et entendre le son. Vous remarquerez peut-être que vous pouvez déclencher le changement en cliquant juste à côté de

l'image. C'est lié au fait que le format réel de l'acteur est un rectangle dont une partie est invisible. Tout le rectangle de l'image est actif, et pas seulement la partie visible. La détection de collision fonctionne avec le rectangle.

Si vous créez vos images, faites en sorte que la partie visible remplisse le mieux possible le rectangle technique, et détournez bien les parties inutiles qui doivent rester invisibles. La détection de collision sera ainsi plus réaliste. Pour les images qui se déplacent rapidement, personne ne remarquera la détection sur les parties invisibles. En revanche, si vos acteurs sont des cercles, la partie inoccupée du rectangle va souvent provoquer une réaction, ce qui nuira un peu au réalisme de l'action.

Animer vos acteurs

La librairie Pygame Zero comporte une fonction permettant d'animer simplement les objets visuels à l'écran, par exemple pour les faire se déplacer automatiquement. C'est un des autres exemples de la façon dont cette librairie simplifie les premiers pas avec le langage Python pour ceux qui viennent du langage Scratch. Normalement, il faut écrire une boucle de répétition dans laquelle vous modifiez successivement les valeurs des coordonnées X ou Y de l'image. Avec Pygame Zero, tout cela est pris en charge d'avance.

Voici un exemple d'utilisation de la fonction **`animate()`** :

```
animate(nebu1, tween='linear', duration=5, pos=
(nebu1.x, -200))
```

Étudions cet appel de fonction en détail :

- Le premier argument est le nom de l'objet animé, c'est-à-dire l'acteur. Ici, c'est un nuage.
- Le second paramètre précise le type d'animation. Vous pouvez choisir de faire accélérer ou ralentir l'animation. Les nombreuses options disponibles sont fournies dans la documentation en ligne de Pygame Zero (j'explique comment y accéder à la fin du chapitre.) Ici, nous utilisons le mode linéaire ; la vitesse d'animation est ainsi constante.

» L'option **`duration`** indique la durée en secondes de l'animation.

» La dernière option **`pos`** désigne la position finale. Vous fournissez les coordonnées entre parenthèses. Il s'agit d'un tuple, comme pour la position du pointeur de souris. Pour la position finale dans le sens horizontal, j'ai choisi dans l'exemple la même position que celle de départ, donc **nebu1.x**. Pour la position en Y, je choisis la valeur -200, ce qui est largement au-dessus du haut de la fenêtre. Le résultat est que l'objet va s'élever verticalement jusqu'à disparaître en haut.

Pour ajouter cette animation, il suffit d'insérer la ligne imprimée en gras dans l'extrait qui suit juste avant la définition de la fonction de tracé. Enregistrez cette version sous le nom *nebu4.py*. Je rappelle que ce qui suit n'est qu'un extrait. Vous devez conserver toutes les autres lignes de la version *nebu3.py* :

Listing 12.3 : Ajout d'une animation (nebu4.py).

```
nebu1 = Actor('target5')
nebu1.x = 250
nebu1.y = 250
animate(nebu1, tween='linear', duration=5, pos=
(nebu1.x, -200))
```

Prenez garde à bien insérer deux parenthèses fermantes à la fin de la nouvelle instruction. Une paire de parenthèses sert à délimiter la totalité des paramètres de la fonction **`animate()`** et le second jeu de parenthèses incarne le tuple des deux coordonnées de l'option **pos**.

Si vous lancez l'exécution de ce programme, vous allez constater que l'objet s'élève verticalement. À chaque appel automatique à la fonction de dessin, l'image est affichée à une position différente, jusqu'à disparaître par le haut de l'écran.

Vous pouvez même cliquer pour faire éclater le nuage. Vous disposez donc dès à présent du moteur minimal d'un jeu vidéo. Ajoutons maintenant une part de hasard pour rendre le jeu plus intéressant.

Utiliser des nombres aléatoires

Nous avons vu dans le Chapitre 11 comment utiliser la fonction prédéfinie **`random. randint()`** pour obtenir un nombre entier à peu près au hasard, dans une plage limitée. Il existe une variante de cette fonction, nommée **`random.uniform()`**, qui permet de faire générer un nombre non entier, donc avec un point décimal (remplace la virgule utilisée en français). Voici par exemple comment faire générer un nombre dit à virgule flottante entre un et trois :

```
random.uniform(1, 3)
```

Vous pouvez tester cette fonction dans la fenêtre de l'interpréteur, c'est-à-dire sans prendre la peine pour trois lignes d'écrire les lignes, de sauvegarder puis de lancer l'exécution. Saisissez les trois lignes suivantes en validant chaque fois par la touche Entrée :

```
>>> import random
>>> for i in range(10):
print(random.uniform(1, 3))
```

N'oubliez pas le signe deux-points à la fin de la deuxième ligne. Automatiquement, l'interpréteur va indenter la ligne suivante, ce qui prouve que Python a compris que vous allez saisir une instruction qui va dépendre de cette boucle de répétition. Une fois cette troisième ligne saisie, ajoutez une ligne vide pour confirmer que la saisie est terminée. Vous devriez alors voir apparaître 10 valeurs numériques dans ce genre de format :

```
2.660707438900764
1.5292264804300049
2.9317931171330924
```

Nous allons donc exploiter quelques nombres aléatoires pour rendre l'animation moins prévisible. Il faut d'abord ajouter tout au début du programme une instruction pour demander d'importer le module *random* dans lequel est définie la fonction dont nous avons besoin :

```
import random
```

Repartez de votre code source *nebu4.py* et modifiez l'appel à la fonction **`animate()`** en ajoutant deux fois un appel à une fonction pour générer un nombre aléatoire. Dans l'extrait qui suit, il s'agit d'une seule ligne :

```
animate(nebu1, tween='linear',
duration=random.uniform(1, 3), pos=(nebu1.x +
random.randint(10, 100), -200))
```

Dans cette nouvelle version, nous choisissons pour la durée une valeur flottante entre une et trois secondes. Lorsque nous avions créé cet exemple, nous avions d'abord essayé avec une valeur entière, mais le résultat était que les nuages semblaient se déplacer de façon peu naturelle, tous à la même vitesse. Avec une valeur flottante entre un et trois, tous les nuages se déplacent à une vitesse différente, et certains vont même en dépasser d'autres, ce qui ajoute au réalisme du jeu.

Pour modifier la position horizontale d'arrivée (dans le sens des X), nous ajoutons une valeur entière au hasard entre 10 et 100. Le résultat est que chaque nuage se décale un peu dans le sens horizontal. Lancez plusieurs fois l'exécution pour voir comment cela varie d'une fois à la suivante. L'effet est subtil, mais il suffit pour augmenter le réalisme lorsqu'il y a plusieurs nuages à l'écran.

Si vous trouvez que le jeu va trop vite, modifiez les nombres qui représentent la plage de l'appel à la fonction de génération de la durée. Le premier nom correspond à la durée minimale, et le second à la durée maximale. Si vous écrivez par exemple **`random.uniform (3,5)`**, vous aurez entre trois et cinq secondes pour attraper chaque nuage. Faites des essais pour trouver la vitesse qui vous convient le mieux.

Ajouter d'autres acteurs

Nous avons vu dans le Chapitre 11 comment créer une liste pour y stocker plusieurs chaînes de caractères. Rien ne nous empêche de stocker dans une liste plusieurs acteurs, ce qui permet de les gérer dans des boucles de répétition. Dans notre jeu, nous allons utiliser six nuages. Dès qu'un des nuages aura disparu par le haut de l'écran, nous allons le faire réapparaître en bas, tant que le chronomètre n'a pas atteint zéro.

Nous allons repartir des idées présentées au début du chapitre, mais l'approche étant différente, mieux vaut partir d'un nouveau fichier vide.

Créez dans l'éditeur un nouveau fichier et donnez-lui le nom *nebu5.py*. Voici la première partie du nouveau code source.

Listing 12.4 : Début de l'exemple nebu5.py.

```
import random

WIDTH  = 500
HEIGHT = 500
score = 0
timer = 10
nebus = list()

for i in range(6):
    nomFic = 'target' + str(i)
    nebus.append(Actor(nomFic))
    nebac = nebus[i]
    nebac.x = random.randint(int(nebac.width /
2), int(WIDTH - nebac.width  ↲
 / 2))
    nebac.y = HEIGHT + nebac.height
    animate(nebac, tween='linear',
duration=random.uniform(1, 3), ↲
pos=(nebac.x + random.randint (10, 100), -200))

def draw():
    screen.clear()
    for i in range(6):
        nebac = nebus[i]
        nebac.draw()

    screen.draw.text("Score: " + str(score), (2,
2), color="red")
    screen.draw.text("Timer: " + str(timer),
(WIDTH-70, 2), color="red")
    if timer == 0:
```

```
        screen.draw.text(str(score), (130, 120),
color="white", fontsize=300)
```

Nous commençons dans le programme par déclarer quatre variables, dont deux nouvelles, ainsi qu'une variable qui va incarner notre liste, en lui donnant le nom **nebus**.

Nous entrons ensuite dans une boucle qui va se répéter six fois, en donnant à la variable **i** tour à tour les valeurs 0 à 5. Cette boucle permet de fabriquer le nom de fichier complet pour chacune des images en mettant bout à bout la partie fixe *target* et la représentation sous forme de chiffre du contenu de la variable **i**, ce qui permet d'obtenir les noms *target0*, *target1*, *etc*. Vous remarquez l'appel à la fonction de conversion **`str()`** ; c'est elle qui convertit la valeur numérique qu'elle trouve dans la variable **i** en un chiffre pouvant s'afficher à l'écran ou s'utiliser dans un nom de fichier. Une fois chaque objet nuage créé grâce aux fichiers d'image, nous disposons d'une liste complète contenant six acteurs.

Nous utilisons ensuite une variable portant le nom **nebac** qui fera référence lors de chaque tour de boucle à un des six acteurs de la liste.

Pour la position en X, nous choisissons une valeur au hasard dans les limites de la largeur de fenêtre. Pour éviter à l'image de disparaître par les côtés, nous utilisons la largeur totale de la fenêtre de laquelle nous soustrayons la moitié de la largeur de l'image. Puisque l'image est centrée, nous sommes certains qu'elle restera entièrement visible en donnant comme limite la moitié de sa largeur. Si par exemple l'image est de 100 pixels de large, son centre ne doit jamais être affiché à moins de 50 pixels du bord gauche. Pour le bord droit, le principe est le même, sauf que nous soustrayons la valeur au lieu de l'ajouter. La position en X qui sera choisie sera une valeur aléatoire entre ces deux limites de gauche et de droite.

Pour la position verticale en Y, nous utilisons la hauteur de la fenêtre à laquelle nous ajoutons la hauteur de l'image. L'image débute donc son existence plus bas que le bas de la fenêtre.

Dès que l'image est positionnée, nous procédons à son animation.

Dans la fonction de dessin **`draw()`**, une boucle permet d'extraire tour à tour chacune des six images. Nous extrayons un acteur de la liste pour le placer dans la variable de travail **nebac** avant d'effectuer l'affichage.

Cette fonction procède ensuite à l'affichage de deux textes dans les coins supérieur gauche et supérieur droit pour le score et le chronomètre. Ces textes ne seront jamais masqués par les nuages, parce qu'ils sont affichés avant. De cette façon, les nuages sembleront passer derrière le texte, ce

qui n’est pas d’un mauvais effet. Pour l’affichage du texte, nous utilisons la méthode **`screen.draw. text()`** en lui donnant en paramètre la chaîne correspondant au texte, les coordonnées X, Y et la couleur d’affichage. Vous pouvez ajouter d’autres options, et notamment la taille des caractères. C’est ce que nous faisons dans le test qui termine la fonction : si le compte à rebours est terminé, nous affichons en plus gros caractères le score obtenu en plein milieu d’écran.

Après avoir saisi ce programme et vérifié qu’il n’y a pas d’erreur, vous pouvez en lancer l’exécution par . Vous devriez voir les six nuages s’élever dans la fenêtre comme dans la Figure 12.2. Tous les nuages commencent au même moment, mais leurs durées d’animation étant différentes, ils vont rapidement s’écarter les uns des autres.

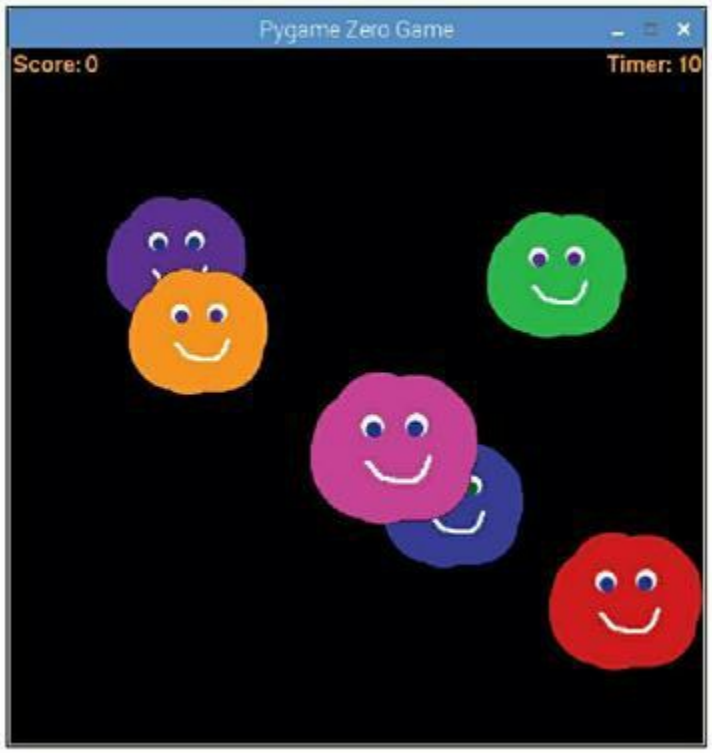


Figure 12.2 : Des nuages qui s’envolent.

Regénérer des nuages

Nous avons bien progressé, mais pour l’instant une fois que les nuages ont disparu en haut de fenêtre, vous ne les voyez plus. Ajoutons la définition de fonction suivante pour qu’un nuage soit généré en bas de fenêtre une fois qu’il a disparu par le haut. Nous ajoutons cette fonction dans le projet *nebu5.py*, après la fonction **`draw()`** que vous nous venons de définir, en laissant bien une ligne d’espace et en commençant dans la marge gauche pour la première ligne.

Listing 12.5 : Fonction update() dans nebu5.py.

```
def update():
    for i in range(6):
        nebac = nebus[i]
        if nebac.y < 0 - nebac.height:
            nebac.y = HEIGHT + nebac.height
             if timer > 0:
nebac.x = random.randint(int(nebac.width / 2),
int(WIDTH - ↲
nebac.width / 2))
nebac.image = 'target' + str(i)
animate(nebac, tween='linear',
duration=random.uniform(1, ↲
3), pos=(nebac.x + random.randint(10, 100),
-200))
```

La fonction de mise à jour nommée **`update()`** est comme la fonction **`draw()`** une fonction automatique de Pygame Zero. Elle est lancée régulièrement sans que vous le demandiez. C'est dans cette fonction que vous réunissez les instructions qui permettent de mettre à jour la position de vos acteurs, et d'actualiser les indicateurs d'avancement dans le jeu. Dans notre exemple, la fonction contient une boucle de répétition qui traite tour à tour chacun des six acteurs en utilisant la variable de travail **nebac**. Lorsque le nuage en cours d'étude a disparu de l'écran, c'est-à-dire que sa position en Y est devenue inférieure à zéro moins sa hauteur, nous pouvons changer cette position Y pour que l'acteur réapparaisse par le bas de la fenêtre. Dans l'exemple, nous prenons même un peu de marge puisqu'il suffirait d'ajouter la moitié de la hauteur de l'image, alors que nous ajoutons la hauteur totale ici. En partant d'encore plus bas comme nous le faisons, les nuages semblent apparaître de façon encore plus aléatoire.

Si le temps n'est pas encore écoulé, nous recalculons une position initiale dans le sens horizontal pour l'objet, sans oublier de le re-raccorder à l'image de nuage de la série *targetx*. En effet, certains des nuages auront été cliqués et seront devenus des images pop. (Le code source qui manque pour cela va être ajouté dans la section suivante.) Nous démarrons alors une nouvelle animation avec la même instruction qu'en début de premier tour.

Une fois que vous avez saisi cette fonction, sauvegardez et lancez l'exécution par . Vous devriez pouvoir admirer un flot continu de nuages ne s'arrêtant qu'à la fin du compte à rebours.

Cliquer sur plusieurs nuages

Dans la première version du projet, nous avons appris à faire éclater un nuage en cliquant, puis comment afficher et animer plusieurs nuages. Combinons ces techniques en définissant une nouvelle version de la fonction qui permet de réagir aux clics souris.

Renommez le code source actuel sous le nom *nebu6.py* puis ajoutez la définition de fonction de gestion des événements souris **`on_mouse_down()`** tout à la fin.

Listing 12.6 : Ajout de on_mouse_down (nebu6.py).

```
def on_mouse_down(pos):
global score
for i in range(6):
    nebac = nebus[i]
    if nebac.collidepoint(pos) and timer > 0 and
nebac.image != 'pop':
        nebac.image = 'pop'
        sounds.blop.play()
        score += 1
```

Rappelons que cette fonction est appelée automatiquement lorsqu'un clic de **souris** a été détecté. Elle contient une boucle qui visite tour à tour chacun des six objets pour savoir s'il a été touché. Un nuage ne doit éclater que si les trois conditions suivantes sont réunies :

1. **le nuage a été cliqué, ce que nous testons par l'appel**.
2. **Le compte à rebours du jeu n'est pas arrivé à zéro, ce qui correspond au test** .
3. **Le nuage n'a pas déjà été éclaté, ce que nous testons en vérifiant que son image actuelle ne**

correspond pas au nom de fichier *pop*, par l'instruction .

Si les trois conditions sont réunies, le programme change l'image du nuage en l'associant à l'image *pop*, fait jouer le son correspondant et augmente le score de un. Une fois que l'image du nuage a changé, il continue à s'élever pour disparaître par le haut de l'écran.

En langage Python, une variable dont la première utilisation se trouve dans une fonction n'est connue que dans cette fonction. C'est une *variable locale*. Ce cantonnement (également appelé *portée*) évite de voir une fonction modifier par mégarde la valeur d'une variable qui est également utilisée par d'autres fonctions. En revanche, une *variable globale* est accessible à toutes les fonctions. Elle est partagée. Nous avons besoin de rendre accessible depuis deux points du programme la variable qui contient le score : dans la fonction qui fait progresser ce score et dans la fonction qui va l'afficher. Pour pouvoir accéder à une variable globale de l'intérieur d'une fonction, il faut déclarer au début de cette fonction que nous allons utiliser une variable globale en écrivant le mot réservé **global** suivi du nom de la variable.

La présente version du programme est tout à fait jouable. Faites un essai. Vous pouvez cliquer sur plusieurs nuages, voir le score augmenter en haut à gauche, et voir de nouveaux nuages apparaître par le bas. Le seul souci, c'est que le jeu ne s'arrête jamais. En effet, nous avons donné à la variable **timer** la valeur 10 en début de programme, mais nous ne la faisons jamais descendre jusqu'à zéro.

Ajouter un chronomètre

Pygame Zero permet de programmer (c'est le cas de le dire) une fonction pour qu'elle soit exécutée de façon périodique ; c'est exactement ce dont nous avons besoin pour notre chronomètre. Nous avions donné la valeur 10 à la variable **timer** en début de programme. Il nous faut prévoir une instruction pour faire diminuer de un cette valeur à chaque appel à la fonction préprogrammée. Il ne reste plus ensuite qu'à faire en sorte que cette fonction soit appelée une fois par seconde. Voici ce qu'il faut rajouter tout à la fin du programme. Il y a une définition de fonction et un appel de fonction isolé tout en bas. Créez le fichier d'étape *nebu7.py* pour cette dernière version :

Listing 12.7 : Ajout du chronomètre (nebu7.py).

```
def krono():
global timer
timer -= 1
if timer == 0:
clock.unschedule(krono)
sounds.whoops.play()

## Programmer la fonction de rappel
clock.schedule_interval(krono, 1)
```

Notez bien que la dernière ligne commence sur la marge gauche après une ligne vide et une ligne de commentaires. Cette instruction ne fait donc pas partie de la fonction définie juste avant et elle est exécutée dès le début du programme. Son premier paramètre est le nom de la fonction qu'elle doit exécuter de façon régulière, c'est-à-dire ici la fonction que nous avons définie juste avant, **`krono()`**. Vous remarquez que nous ne faisons exceptionnellement pas suivre le nom de la fonction de son couple de parenthèses. Le deuxième paramètre indique le nombre d'appels par seconde, ici un. Utilisée ainsi, notre fonction **`krono()`** devient une fonction de rappel (*callback*). Ce n'est pas vous qui l'appelez, mais le moteur d'exécution de Python.

Revenons justement à cette fonction **`krono()`**. Elle commence par déclarer qu'elle va utiliser une variable globale nommée **timer** puis elle décrémente cette variable (elle diminue sa valeur de 1) et teste si la variable est devenue égale à zéro. Dans ce cas, la partie est finie, et la fonction arrête l'appel automatique à elle-même avec **`clock.unschedule()`** puis elle fait jouer le son de fin de jeu. En revanche, tant que la valeur de **timer** n'est pas nulle, la fonction n'a qu'un seul effet : soustraire un au compte à rebours.

Si vous lancez l'exécution de cette version, vous devez voir le chronomètre descendre jusqu'à zéro dans le coin supérieur droit. Dès qu'il atteint zéro, les nuages n'apparaissent plus par le bas. Ceux qui étaient déjà visibles continuent à monter jusqu'à disparaître puis le score est affiché en grand format au milieu de l'écran.

Ajuster la difficulté du jeu

Voyons pour finir comment intervenir sur la plus ou moins grande difficulté du jeu :

» Vous pouvez intervenir sur la taille ou la forme de la fenêtre en jouant sur les deux constantes WIDTH et HEIGHT. Si la fenêtre est plus large, les nuages sont moins faciles à viser. Si elle est plus haute, vous avez plus de temps pour cliquer.

» Vous pouvez intervenir sur la vitesse des nuages en revoyant la durée calculée dans les appels à la fonction **`animate()`**. Rappelons qu'il y a un appel à cette fonction au démarrage et un autre dans la fonction de mise à jour **`update()`**.

Pour faciliter le jeu, il suffit d'augmenter la valeur de départ du chronomètre.

Listing complet du code source

Voici pour finir le code source complet du projet (Listing 12.8).

Listing 12.8 : Code source complet du jeu nebulos.py.

```
# nebulos (VF de Cloudbusting)
# VO Sean McManus - www.sean.co.uk
# VF O. Engler
# Pour lancer le jeu :  pgzrun nebulos.py

import random

WIDTH  = 500
HEIGHT = 500
score  = 0
timer  = 10
nebus = list()

for i in range(6):
    nomFic = 'target' + str(i)
```

```
   nebus.append (Actor(nomFic))
    nebac = nebus[i]
    nebac.x = random.randint(int(nebac.width /
2), int(WIDTH - nebac.width ↲
/ 2))
    nebac.y = HEIGHT + nebac.height
    animate(nebac, tween='linear',
duration=random.uniform(1, 3), ↲
pos=(nebac.x + random.randint(10, 100), -200))

def draw():
    screen.clear()
    for i in range(6):
        nebac = nebus[i]
        nebac.draw()

    screen.draw.text("Score: " + str(score), (2,
2), color="red")
    screen.draw.text("Timer: " + str(timer),
(WIDTH - 70, 2), color="red")
    if timer == 0:
        screen.draw.text(str(score), (130, 120),
color="white", font ↲
size=300)

def update():
    for i in range(6):
        nebac = nebus[i]
        if nebac.y < 0 - nebac.height:
            nebac.y = HEIGHT + nebac.height
             if timer > 0:
                nebac.x =
random.randint(int(nebac.width / 2), int(WIDTH -
↲
nebac.width / 2))
                nebac.image = 'target' + str(i)
```

```
                    animate(nebac, tween='linear',
duration=random.uniform(1, ↲
3), pos=(nebac.x + random.randint(10,  100),
-200))

def on_mouse_down(pos):
    global score
    for i in range(6):
        nebac = nebus[i]
        if nebac.collidepoint(pos) and timer > 0
and nebac.image != 'pop':
            nebac.image = 'pop'
            sounds.blop.play()
            score += 1

def krono():
    global timer
    timer -= 1
    if timer == 0:
        clock.unschedule(krono)
        sounds.whoops.play()

##
clock.schedule_interval(krono, 1)
```

Pour aller plus loin dans Pygame Zero

Nous espérons que ce projet vous a donné une idée de ce qu'il était possible de faire avec Pygame Zero. Les fonctions disponibles permettent de détecter des frappes de touches, ce qui autorise la création de jeux dans lesquels le personnage est contrôlé au clavier. Beaucoup de jeux sont bâtis sur ce genre d'interaction.

Voici quelques pistes pour en savoir plus sur Pygame Zero :

» Le blog de Daniel Pope qui annonce Pygame Zero :

http://mauveweb.co.uk/posts/2015/05/pygar
zero.html

» La documentation de Pygame Zero avec des exemples :

http://pygame-
zero.readthedocs.io/en/latest

» La liste des objets prédéfinis de Pygame Zero (acteurs, images, sons, horloge) :

http://pygame-
zero.readthedocs.io/en/latest/builtins.h

Chapitre 13

Programmer Minecraft en Python

DANS CE CHAPITRE :

- **Découvrir le monde Minecraft**
- **Modifier Minecraft en Python**
- **Générer un labyrinthe dans Minecraft**

Minecraft est ce jeu vidéo de construction d'univers qui a attiré à lui tous les amateurs de briques Lego. Il permet en effet de construire de vrais mondes en trois dimensions à partir de blocs cubiques. Sa capacité à stimuler l'imagination est telle que plus de 100 millions d'exemplaires du jeu ont été vendus, sur des plates-formes très diverses, du PC à la console Xbox.

Il existe même une version de Minecraft pour le Raspberry Pi. Elle ne permet que de travailler en mode création. Vous ne serez donc ni gêné par les monstres, ni angoissé de voir diminuer vos ressources. Cette version peut être programmée en utilisant différents langages, parmi lesquels le langage Python. Vous pouvez ainsi créer un château géant sans devoir positionner les blocs un à un. Il va devenir possible de créer des structures originales puis de se promener à l'intérieur, comme nous allons le voir dans ce chapitre.

Nous proposons en effet de rédiger le code source d'un programme Python pour construire automatiquement un labyrinthe dans Minecraft. À chaque lancement du programme, le labyrinthe sera différent et vous pourrez paramétrer sa taille et les matériaux utilisés pour les murs. Nous en profiterons pour découvrir comment placer et supprimer des blocs en langage Python. Vous allez ainsi acquérir des compétences pour pouvoir ensuite écrire d'autres programmes de construction.

La version actuelle de Minecraft Pi Edition est en version alpha, c'est-à-dire une version très préliminaire, mais les défauts que contient cette version restent largement supportables. Nous avons par exemple remarqué que le pointeur se comportait bizarrement lorsque nous agrandissions la fenêtre. Vous verrez aussi un défaut d'alignement entre le contenu de la fenêtre et son cadre. Il n'y a pas non plus de son.

J'ai indiqué dans l'introduction comment récupérer les fichiers de code source de ce chapitre. Je fournis le code de l'exemple complet à la fin du chapitre. En le parcourant, vous pourrez revoir les différentes techniques abordées au cours du chapitre.

Découvrir le monde Minecraft

Le jeu Minecraft est préinstallé dans Raspbian. Vous pouvez le démarrer directement en ouvrant le menu **Applications** puis la catégorie des jeux **Games**. Dès que vous démarrez le jeu, vous arrivez à l'écran de titre qui offre deux possibilités :

- **Start game :** lance la génération d'un monde à explorer. Ce choix permet également de désigner un monde à recharger. Vous choisissez parmi les différents mondes trouvés en cliquant dans le monde désiré pour l'amener au milieu, puis en cliquant à nouveau pour l'ouvrir.
- **Join game** : cette option permet de se connecter à d'autres joueurs dans un réseau local. Je ne décris pas cette option dans ce chapitre, mais c'est elle qui permet de travailler ou de jouer à plusieurs dans un monde Minecraft.

Choisissez donc l'option **Start game** puis **Create new**. Minecraft va générer un nouveau monde avec des montagnes, des forêts et des océans. Laissez-le travailler. Vous voyez ensuite ce monde apparaître en vue subjective (Figure 13.1).

Vous pouvez changer de point de vue afin de voir le personnage dans le jeu. Utilisez la touche Echap pour revenir au menu du jeu, puis cliquez l'icône qui est à côté de celle du haut-parleur en haut à gauche.

Quand vous avez terminé de vous promener ou de construire, sortez du jeu en frappant la touche Echap pour revenir au menu du jeu, puis en choisissant **Quit to Title**.

Figure 13.1 : Premier contact visuel avec le monde Minecraft sur Raspi.

Se déplacer

Minecraft est facile à jouer en utilisant une main pour la souris et l'autre pour se déplacer au clavier. La souris sert à contrôler l'orientation de votre regard, et donc à tourner à droite et à gauche. Pour vous déplacer, vous utilisez les quatre touches W, S, A et D. Les touches W et S servent à avancer et à reculer et les touches A et D à se décaler vers la gauche et la droite. Sur un clavier anglais, ces quatre touches forment une croix et tombent bien sous les doigts.

N. d. T : Sur un clavier français, les touches pour avancer et reculer W et A ne sont pas pratiques. Si vous voulez retrouver la même configuration physique que sur un clavier américain, il suffit, avant de lancer Minecraft, de basculer votre clavier en configuration anglaise au moyen de la commande suivante dans une fenêtre de terminal :

```
setxkbmap us
```

Pour revenir au clavier français, vous saisissez la commande suivante :

```
setxkbmap fr
```

Votre personnage grimpe sans souci sur les blocs d'une seule unité de hauteur, mais vous pouvez utiliser la barre d'espace pour sauter.

Pour obtenir un bon aperçu de votre monde, il suffit de vous envoler en tapant deux fois la barre d'espace. Pour monter encore plus, maintenez enfoncée la barre d'espace. Pour redescendre, vous utilisez la touche MAJ de gauche. Pour retomber au sol, tapez deux fois la barre d'espace. Dans ce mode création, vous ne risquez pas de vous blesser. Vous pouvez donc retomber de très haut sans problème.

Créer et casser des blocs

Pour casser un bloc, il suffit de pointer vers ce bloc avec la souris puis de cliquer avec du bouton gauche. Vous verrez que certains blocs sont plus faciles à casser que d'autres. Vous devez également vous approcher suffisamment. Si vous ne voyez pas le bloc commencer à se détruire, c'est que vous êtes trop loin.

Le panneau qui se trouve dans le bas de la fenêtre présente une sélection de blocs (revoyez le bas de la Figure 13.1). Pour choisir une des matières disponibles, vous utilisez la molette de la souris ou vous tapez directement un nombre entre un et huit (de gauche à droite). Pour accéder à votre inventaire complet, vous utilisez la touche E. Vous pouvez alors vous déplacer parmi les différents blocs et en choisir un avec la touche Entrée. Vous pouvez également directement cliquer le bloc.

Pour placer un bloc, vous cliquez droit à l'endroit désiré. Vous ne pouvez poser un bloc sur un autre que si vous voyez le dessus du bloc précédent. Voilà pourquoi vous aurez parfois besoin de vous envoler, notamment pour construire des bâtiments hauts.

Pour construire rapidement une tour, il suffit de regarder vers le bas en s'envolant puis de sauter à répétition tout en plaçant un bloc à chaque fois sous ses pieds.

Nous allons automatiser la construction d'éléments grâce au langage Python, mais nous vous conseillons de prendre d'abord un peu de temps pour voir comment construire sans l'aide d'un langage. Regardez notamment comment les différentes matières de blocs interagissent. Par exemple, un bloc de pierre peut être posé dans l'air, mais un bloc de sable retombe au sol. Vous ne pouvez pas déposer de blocs de cactus dans de l'herbe, mais vous pouvez les planter dans du sable. Si vous détruisez les berges d'un lac, l'eau va s'écouler et provoquer une inondation. Dans cette version, vous ne pouvez pas déposer des blocs d'eau ou de lave dans

le jeu, mais vous pouvez les faire apparaître grâce au langage Python. Les liquides vont alors descendre vers le point le plus bas et noyer toute une région. Lorsque l'eau et la lave entrent en contact, il arrive que l'eau soit en volume suffisant pour refroidir la lave et la transformer en pierre.

Préparer Python pour Minecraft

Une des caractéristiques spéciales du jeu Minecraft est qu'il prend entièrement le contrôle du pointeur de souris. Pour pouvoir accéder aux autres fenêtres, vous devez frapper la touche Tab. Pour revenir à Minecraft ensuite, il suffit de cliquer dans les limites de la fenêtre du jeu.

Vous allez donc prendre l'habitude d'utiliser la touche Tab. Justement, frappez maintenant Tab pour sortir de la fenêtre de Minecraft et revenir au bureau. Pour rédiger le texte source de votre programme, nous allons utiliser un des éditeurs disponibles pour Python. Ouvrez le menu **Applications**, la catégorie **Programmation** et choisissez soit **Thonny**, soit **Python 3 IDLE**. Vous aurez sans doute besoin de repositionner la fenêtre de Minecraft pour pouvoir voir celle de l'éditeur. Vous pouvez même la replier en cliquant dans le premier des trois boutons de l'angle supérieur droit de la fenêtre.

Vous allez vite remarquer que la fenêtre de Minecraft veut toujours rester au premier plan. La fenêtre de votre éditeur Python va donc être cachée derrière. Il va falloir faire un peu de ménage. Si votre écran est assez grand, vous allez pouvoir positionner côte à côte les fenêtres de Minecraft et de votre éditeur.

Si l'écran n'est pas assez grand, il faudra passer d'une fenêtre à l'autre. Pour masquer la fenêtre de Minecraft, il suffit de cliquer dans sa case de la barre des tâches puis dans le bouton de la barre de titre de la fenêtre. Pour déplier Minecraft, il suffit de cliquer à nouveau dans la barre des tâches dans son icône. Si vous ne voyez pas la barre de titre de Minecraft, c'est qu'elle ne va pas réagir à vos actions à la souris au clavier. Il faut dans ce cas cliquer dans la barre des tâches. (Nous avons vu dans le Chapitre 4 comment bien naviguer sur le bureau.)

Le module Minecraft

Pour notre premier programme Python de contrôle de Minecraft, voyons comment faire afficher un message dans la fonction de tchat du jeu.

Vous commencez par saisir dans l'éditeur de votre environnement les quatre instructions de l'extrait suivant. Vous sauvegardez ensuite le tout

dans un fichier dans votre dossier *pi*, par exemple sous le nom *mctest.py*. Lancez ensuite l'exécution avec le bouton **Run** ou la touche F5. Si la fenêtre de Minecraft avait été refermée, il faut redémarrer le jeu, c'est indispensable.

```
import sys, random
from mcpi  import minecraft
mc = minecraft.Minecraft.create()
mc.postToChat("Bienvenue dans mon MineLabyr !")
```

La première ligne demande d'importer le contenu des deux librairies **`sys`** et **`random`**. Nous n'avons pas immédiatement besoin de la seconde, mais elle sera nécessaire pour construire le labyrinthe avec des valeurs aléatoires plus tard.

Pour écrire en langage Python une instruction qui va contrôler le comportement de Minecraft, il faut d'abord utiliser la fonction **`minecraft. Minecraft. create()`**, et ajouter le nom de la commande à déclencher à la suite de cette déjà longue ligne (avant les parenthèses). Voici comment il faudrait normalement écrire la commande pour afficher un message dans la fenêtre de tchat avec la fonction **`postToChat()`** :

```
minecraft.Minecraft.create().postToChat("Bienvenue
 dans mon MineLabyr !")
```

C'est pour éviter cette longue instruction que nous demandons dans la troisième ligne de créer un objet portant le nom **mc**, qui va remplacer par deux caractères les 26 caractères de **`minecraft. Minecraft.create`**.

Si le programme ne fonctionne pas, vérifiez d'abord la différence entre les majuscules et minuscules, c'est-à-dire la casse des lettres. En effet, le langage Python distingue les majuscules des minuscules. Faites notamment attention au M majuscule dans **`minecraft. Minecraft.create`**.

Dès que vous avez lancé l'exécution, rebasculez vite dans la fenêtre de Minecraft pour avoir le temps de voir apparaître le message, car il disparaît après 10 secondes environ.

Les coordonnées spatiales dans

Minecraft

Puisque Minecraft est une représentation d'un monde en trois dimensions, il est logique que tout ce qui s'y trouve soit défini par des coordonnées, selon trois axes : largeur, longueur et hauteur :

- **X** : premier axe horizontal. Le monde mesure 255 blocs dans cette orientation.
- **Y** : axe vertical qui correspond à l'altitude. Vous pouvez vous élever au moins jusqu'à 500 blocs de haut, mais vous ne voyez plus le sol à partir de 70 blocs environ, il n'y a donc pas d'intérêt à monter si haut. Le niveau de la mer correspond bien sûr au niveau zéro. Vous pouvez même descendre plus bas en creusant. Nous avons ainsi réussi à descendre jusqu'à – 70, mais à ce moment-là, nous sommes morts. C'est d'ailleurs la seule manière de succomber dans la version Pi de Minecraft.
- **Z** : second axe horizontal qui est limité lui aussi à 255 blocs.

Si l'axe des X correspond aux déplacements en largeur, à gauche et à droite, l'axe des Z correspond aux déplacements en avant et en arrière.

Ce n'est pas par hasard que nous avons présenté l'axe de l'altitude avant le second axe horizontal. C'est en effet dans cet ordre que Minecraft doit recevoir les coordonnées. Dans un univers en 2D, tel que celui de Scratch, nous ne spécifions que deux coordonnées, X et Y. Il faut donc un peu de temps pour s'habituer au fait que Y correspond dans Minecraft à l'altitude. Dans le générateur de labyrinthe que nous allons créer, nous allons sans cesse faire varier les coordonnées X et Z pour désigner la position de chaque brique du mur, alors que la coordonnée d'altitude Y va toujours rester la même.

Quand vous jouez, vous pouvez voir les coordonnées du joueur dans le coin supérieur gauche de la fenêtre de Minecraft. Si vous tentez de sortir des limites du jeu, vous allez entrer en collision avec un mur invisible, un

peu comme dans le film *The Truman Show*, sauf qu'il n'y a ici pas de porte.

Positionner le joueur

Vous pouvez replacer votre personnage à n'importe quel endroit dans le monde de Minecraft au moyen de la commande suivante :

```
mc.player.setTilePos(x, y, z)
```

Voici par exemple comment apparaître dans le jeu en descendant en parachute :

```
mc.player.setTilePos(0, 100, 0)
```

Vous n'êtes pas obligé d'ajouter cette instruction dans le programme que vous venez d'écrire pour la tester. À partir du moment où vous avez lancé au moins une fois le début de programme qui importe le module Minecraft, vous pouvez saisir directement des commandes pour déplacer le joueur et ajouter des blocs dans la fenêtre de l'interpréteur Python.

Lorsque vous exécutez cette commande **`setTilePos()`**, et à condition que vous ne soyez pas en mode volant, vous allez tomber du ciel. Si vous êtes en mode volant, frappez deux fois la barre d'espace pour chuter.

Sachez que les coordonnées indiquées ne vont pas toujours correspondre au milieu du monde, même si tous les mondes ont la même taille. Lors d'un de nos essais, les coordonnées réelles allaient de – 85,7 à +169,7 dans le sens des X, mais de – 90,7 à +156,8 dans le sens des Z.

Puisque vous pouvez positionner le joueur n'importe où, il arrivera qu'il surgisse emprisonné dans une montagne ou une autre structure solide, et qu'il ne puisse plus bouger. Dans ce cas, repositionnez le joueur en modifiant l'instruction. En général, il suffit de le faire démarrer assez haut pour qu'il ne se trouve prisonnier d'aucun élément bloquant. Rappelons qu'en mode création, la chute ne peut pas être fatale.

Ajouter des blocs avec setBlock()

Pour ajouter un bloc à votre monde, vous utilisez la commande suivante :

```
mc.setBlock(x, y, z, IDMateriau)
```

Le quatrième paramètre, **`IDMateriau`**, est un nombre qui désigne la matière du bloc. Toutes les valeurs entre 0 et 108 sont reconnues, et même quelques autres valeurs supérieures. Le Tableau 13.1 propose quelques-unes des matières les plus utilisées. Vous trouverez la liste complète sur le site Wiki de Minecraft à l'adresse suivante :

`www.minecraftwiki.net/wiki/Data_values_(Pocket_Edition)`

Tableau 13.1 : Sélection de matériaux dans Minecraft Pi.

0	Air
1	Pierre
2	Herbe
3	Poussière
5	Planche de bois
8	Eau
10	Lave
12	Sable
20	Brique de verre
24	Grès
41	Brique en or
45	Brique
47	Étagère de livres
53	Escalier en bois
57	Diamant
64	Porte en bois
81	Cactus

Si vous comptez utiliser des blocs d'eau ou de lave, vous allez en général noyer le monde. Mieux vaut dans ce cas en créer un nouveau.

Le même nom de commande écrit au pluriel (**`setBlocks()`**) permet de créer toute une structure en une seule fois, avec un seul type de matériau. Il suffit de lui indiquer les coordonnées de deux coins opposés du parallélépipède dans chacune des trois dimensions, comme ceci :

```
mc.setBlocks(x1, y1, z1, x2, y2, z2, blockTypeId)
```

Pour créer du vide dans Minecraft, l'astuce consiste à positionner des blocs de matière Air. On peut facilement créer un petit bâtiment en le remplissant d'abord avec une matière solide puis en l'évidant, sauf les murs en ajoutant de l'air, comme dans cet exemple :

```
mc.setBlocks(0, 0, 0, 10, 5, 7, 45) # Brique
mc.setBlocks(1, 0, 1,  9, 5, 6,  0) # Air
```

La première instruction construit un bâtiment mesurant 10 blocs sur 7 en surface et 5 blocs de haut à partir des coordonnées 0,0,0. Grâce à la seconde instruction, le bâtiment aura des murs d'un seul bloc d'épaisseur parce que nous remplissons tout l'intérieur de 1 à 9 en X, et de 1 à 6 en Z avec des blocs Air (et de 5 à 0 en Y vertical). Il ne restera donc plus qu'un bloc de briques pour constituer les quatre murs, le toit étant rouvert.

Si la fenêtre de Minecraft devient toute noire quand vous lancez cet exemple, c'est que vous avez par malchance fait construire les murs par-dessus le joueur. Dans ce cas, repositionnez le joueur avec l'instruction que vous savez utiliser.

Le signe dièse à la fin des deux dernières instructions représente le début d'un commentaire. Le compilateur ignore ce symbole et tout ce qui le suit jusqu'à la fin de ligne.

Les coordonnées d'un joueur peuvent être exprimées de manière précise avec une partie décimale, comme dans 1.7, mais les blocs sont toujours positionnés après arrondi de leurs coordonnées au nombre entier le plus proche.

Interdire les modifications

Nous savons bien que vous n'allez pas tricher en cherchant à sortir du labyrinthe par destruction des murs, n'est-ce pas ? Pour que le joueur ne

puisse ni détruire ni ajouter des blocs, il faut ajouter la commande suivante :

```
mc.setting(“world.immutable”, True)
```

Le mot *immutable* est souvent utilisé en programmation. Il signifie non modifiable, immuable.

Nous savons maintenant positionner le joueur, déposer des blocs et les supprimer avec le bloc Air. Programmons la construction du labyrinthe.

Définir les paramètres du labyrinthe

Pour notre projet, nous allons définir un certain nombre de constantes qui vont servir à mémoriser des paramètres. Une constante est une variable que vous avez décidé de ne jamais modifier une fois que le programme a démarré.

Par convention, on utilise une écriture en lettres capitales pour les noms des constantes, ce qui permet de les distinguer des variables et de ne jamais oublier que vous avez choisi de ne pas modifier leurs valeurs. En remplaçant des nombres par des constantes qui les symbolisent, vous simplifiez les modifications de votre programme et vous augmentez la lisibilité du code source.

Comme déjà dit, le langage Python distingue les majuscules des minuscules. Les deux mots **ville** et **Ville** sont pour Python deux variables différentes. Vous éviterez bien sûr d’utiliser ce genre d’homonymes, car les humains ne distinguent pas toujours bien la casse de lettres.

Voici donc le début du programme qui consiste à définir un certain nombre de constantes :

```
DIM    = 10
HAUT   = 2
SOL    = 0
LABYX = 0
LABYZ = 0
LABY_MATER = 1 # Pierre
SOL_MATER  = 2 # Herbe
TOIT   = False
```

Pour construire notre labyrinthe, nous faisons d'abord générer une grille de'un bloc de large, un peu comme une gaufre (Figure 13.2). Chacune des cellules comporte donc quatre murs. Le programme va supprimer certains murs pour constituer le labyrinthe. C'est un labyrinthe carré dont la largeur et la longueur sont définies par la constante DIM. Avec une valeur de 10, le labyrinthe contiendra au départ 10 cases dans le sens des X et 10 dans celui des Z, mais il va occuper le double d'espace, c'est-à-dire 20 blocs sur 20 parce qu'il y a un mur d'un bloc de large autour de chaque case. Cela va devenir plus clair lorsque vous aurez commencé à écrire le programme. Nous avons fait des essais avec des labyrinthes allant jusqu'à 40 de côté, mais le temps de génération commence à devenir vraiment long. Pour l'instant, contentons-nous de 10 cases sur 10.

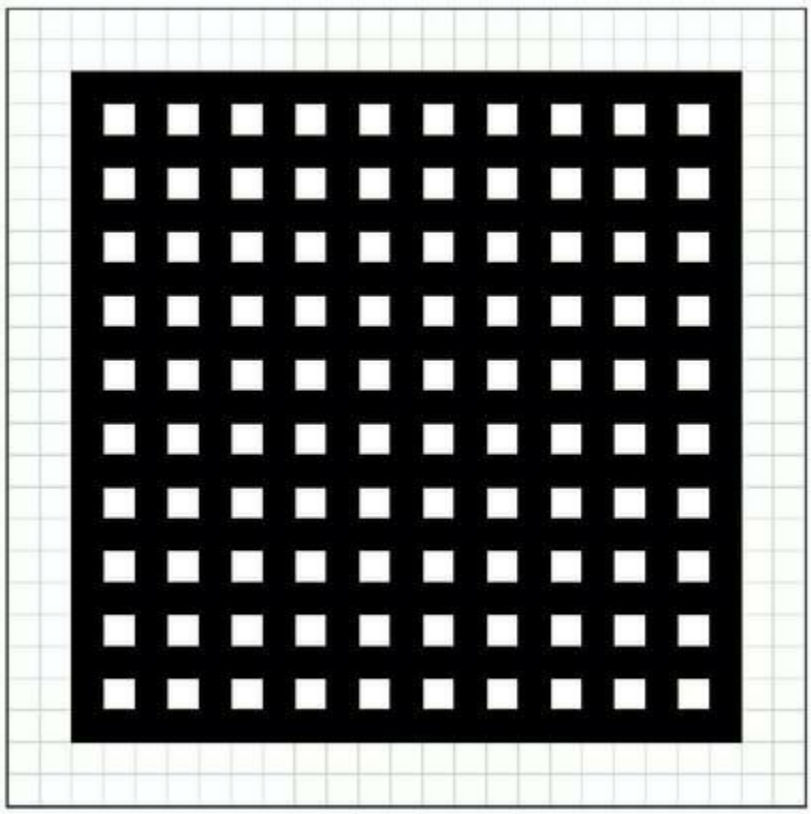


Figure 13.2 : Grille initiale du labyrinthe.

La constante HAUT définit le nombre de briques empilées en hauteur. Nous avons choisi la valeur qui offre le meilleur compromis : une valeur de un laisse le joueur passer par-dessus les murs. (En effet, une hauteur d'une brique de différence n'est pas un obstacle dans ce jeu.) Une valeur supérieure à deux empêcherait de voir les montagnes et le paysage qui constituent de bons repères pour se situer dans le dédale.

Les trois constantes LABYX, SOL et LABYZ déterminent les coordonnées de départ de la construction. Le matériau choisi est la pierre (valeur 1) et celui du sol est l'herbe (valeur 2). Nous avons prévu d'ajouter un toit pour empêcher le joueur de sortir du labyrinthe en

s’envolant. Mais pendant la phase de construction et de mise au point, nous désactivons la construction du toit pour que vous puissiez juger de la qualité du labyrinthe pendant sa construction.

Le programme s’arrête avec une erreur s’il n’y a pas assez de place pour construire la totalité du labyrinthe. Dans ce cas, soit vous réduisez la taille, soit vous modifiez les coordonnées X et Z, soit vous demandez la génération d’un nouveau monde.

Vous pouvez essayer de construire un labyrinthe avec des piles de livres (valeur 47), c’est du plus bel effet !

Poser les fondations

Avant de commencer la construction du labyrinthe, il faut vérifier que vous allez construire sur du solide. Les mondes Minecraft sont générés de façon plus ou moins aléatoire et vous pourriez très bien vous lancer dans la construction d’un labyrinthe dans une montagne ou en pleine mer.

Il faut donc vider non seulement la surface que va occuper le labyrinthe, mais également les alentours sur dix blocs de large pour que le joueur puisse facilement s’approcher et faire le tour du bâtiment. Pour ce faire, vous remplissez la zone avec des blocs de matière Air puis vous ajoutez le sol au moyen d’une couche de blocs avec le matériau choisi pour le sol.

Le labyrinthe va occuper au sol un espace qui s’exprime en blocs entre LABYX et LABYX + (DIM * 2) dans le sens des X, et entre LABYZ et LABYZ+ (DIM * 2) dans le sens des Z. Je rappelle que l’astérisque est le symbole de la multiplication. Le nombre de blocs est le double du nombre de cases exprimées par la constante DIM, puisque chaque case est flanquée d’un mur sur sa droite et d’un autre sur un autre côté. Le milieu du labyrinthe dans le monde de Minecraft correspond à la LABYX+DIM et LABYZ+DIM.

Nous devons donc supprimer 10 blocs de plus dans les deux directions. L’extrait suivant supprime tout matériau jusqu’à 150 blocs de hauteur par rapport au niveau du sol pour éviter à des blocs d’une montagne de retomber dans le labyrinthe. La deuxième instruction met en place le plancher.

N. d. T : Les deuxième et troisième lignes de chaque instruction ci-après sont factices ! Elles ont été créées pour rendre la lecture plus aisée, mais tout doit être saisi à la suite.

```
mc.setBlocks(LABYX-10, SOL,
```

```
                LABYZ-10, LABYX+(DIM*2)+10,
                SOL+150, LABYZ+(DIM*2)+10, 0)
mc.setBlocks(LABYX-10, SOL,
                LABYZ-10, LABYX+(DIM*2)+10,
                SOL, LABYZ+(DIM*2)+10, SOL_MATER)
```

Nous conseillons d'ajouter un bloc dans le coin de démarrage, celui qui correspond à LABYX et LABYZ. Cela vous servira pour la mise au point, parce que vous pourrez vous repérer dans le labyrinthe en le survolant. Voici comment ajouter ce repère :

```
mc.setBlock(LABYX, SOL+HAUT+1, LABYZ, LABY_MATER)
```

Il ne reste plus qu'à positionner le joueur au-dessus du milieu du labyrinthe pour qu'il puisse en admirer la construction en regardant vers le bas. Si vous ne volez pas, vous allez tomber dans le labyrinthe, mais il suffira alors de vous envoler à nouveau :

```
mc.player.setTilePos(LABYX+DIM, SOL+25,
LABYZ+DIM)
```

Monter les murs

Voici le bloc de répétition qui permet de construire les murs de la gaufre initiale :

```
for line in range(0, (DIM+1)*2, 2):
    mc.setBlocks(LABYX+line, SOL+1, LABYZ,
                 LABYX+line, SOL+HAUT, LABYZ+
(DIM*2), LABY_MATER)
    mc.setBlocks(LABYX, SOL+1, LABYZ+line,
                 LABYX+(DIM*2), SOL+HAUT,
LABYZ+line, LABY_MATER)
```

Dans la boucle **`for`**, la variable de boucle **line** va prendre successivement toutes les valeurs paires entre zéro et la valeur qui correspond à DIM fois 2. Vous remarquez qu'il faut ajouter un à la valeur de DIM avant de la multiplier par deux parce que la fonction **`range()`** n'englobe pas dans son traitement la dernière valeur de la plage qui lui est fournie. Si vous indiquez par exemple **`range (1,10)`**, ne seront utilisés que les

nombres entre un et neuf. La valeur 2 qui est en troisième paramètre de notre exemple d'utilisation de `range()` indique la taille du pas de progression. La valeur de la variable augmente donc de deux à chaque tour de boucle, et ne traite donc que les nombres pairs. C'est grâce à cette astuce que chaque case est séparée de la suivante par un mur. À chaque tour de boucle, nous montons deux murs, un dans le sens des X, et l'autre dans le sens des Z.

Bien sûr, dans certains cas, le même bloc est déposé deux fois, là où deux lignes sont en intersection. Nous commençons à monter les murs à partir de la hauteur SOL+1 pour que l'herbe reste intacte, et que vous puissiez la retrouver lorsque vous allez détruire les murs pour sculpter le labyrinthe.

Dans le précédent exemple, il n'y a que trois instructions. Il ne faut surtout pas oublier le signe deux-points à la fin de la première ligne. Veillez aussi à ce que chacune des deux instructions suivantes soit précédée de quatre espaces, ce qui permet à Python de savoir qu'elles doivent être exécutées dans le corps de la boucle.

Lorsque vous lancez l'exécution, vous devriez obtenir une grille non sculptée, comme dans la Figure 13.3.

Figure 13.3 : Aspect initial de la grille dans Minecraft.

L'algorithme de création du labyrinthe

Avant de nous plonger dans l'écriture du code qui va transformer notre gaufre en un beau labyrinthe, voyons les choses au niveau théorique. Nous allons découvrir comment construire un labyrinthe parfait ; ce terme est consacré puisqu'on dit qu'un labyrinthe est parfait lorsqu'il ne contient aucune boucle et que tous les endroits du labyrinthe peuvent être visités. Il n'y a donc qu'un seul chemin entre deux lieux du labyrinthe.

Un algorithme est un jeu de règles ou la description d'un processus qui permet de résoudre un problème. Les algorithmes sont devenus très célèbres depuis l'apparition des ordinateurs, mais vous pouvez dessiner le labyrinthe avec du papier et un crayon avec le même algorithme, en gommant les murs à abattre. Voici cet algorithme :

1. **Au départ de l'algorithme, le bâtiment en forme de gaufre a déjà été généré**. Chaque case est entourée de quatre murs.
2. **Vous choisissez où vous le voulez la case de départ**.
3. **Vous analysez les cases voisines de la case courante pour établir la liste de celles dont les quatre murs sont intacts**. Ce sont les cases non encore visitées/vues.
4. **Vous choisissez une des cases non vues au hasard, vous cassez la cloison qui la sépare de la case courante (là où vous êtes) puis vous entrez dans cette case, qui devient la nouvelle case courante**.
5. **Si la case courante n'a plus de voisine non visitée, vous reculez d'une case sur le chemin mémorisé et vous décrétez cette dernière case courante**.
6. **Répétez les Étapes 3 à 5 jusqu'à avoir visité toutes les cases**.

Définition des variables simples et des listes

Nous avons besoin, en plus des constantes déjà définies, de plusieurs variables :

- **qCasesTot :** mémorise le nombre total de cases du labyrinthe carré, donc DIM * DIM.
- **qCasesVues :** mémorise le nombre de cases déjà vues. Lorsque ce compteur atteint la même valeur que **qCasesTot**, c'est que toutes les cases ont été vues et qu'au moins un mur a été démoli pour chacune, ce qui rend la case accessible. La construction est achevée.
- **posX :** coordonnée en X de la position actuelle, mesurée en cases et commençant par une valeur aléatoire entre 1 et DIM.
- **posZ :** coordonnée en Z de la position actuelle, mesurée en cases et commençant comme celle des X par une valeur aléatoire.
- **listeCasesVues[] :** cette liste mémorise les étapes successives de votre progression dans le dédale, pour pouvoir revenir en arrière. Au départ, vous y stockez les coordonnées initiales avec la méthode **append( )**.
- **joueurX** et **joueurZ** : ces variables stockent la position initiale du joueur pour l'y placer dès que le labyrinthe est prêt.

Ce genre d'algorithme (générateur de labyrinthe à parcours en profondeur) a besoin d'une structure de données de type liste pour mémoriser les coordonnées des murs. Dans le cas particulier de Minecraft, cette liste est incarnée par la construction que nous y ajoutons. La structure de données est rendue visible !

Voici les instructions de préparation des variables globales :

```
qCasesTot  = DIM * DIM
qCasesVues = 1          # 1 pour la case initiale
listeCasesVues = []
posX = random.randint(1, DIM)
posZ = random.randint(1, DIM)
joueurX = posX
joueurZ = posZ
montrerBloc(posX, posZ)

# Voir texte qui suit
listeCasesVues.append((posX, posZ))
```

Définition des fonctions

Nous allons avoir besoin de plusieurs fonctions utilitaires pour traduire notre algorithme en code source :

- **`realx(x)`** et **`realz (z)`** : conversion des coordonnées exprimées en cases de labyrinthe vers les coordonnées réelles de Minecraft (exprimées en blocs et dépendantes de la position de début dans le labyrinthe).
- **`montrerBloc (x, z)`** et **`cacherBloc (x, z)`** : ce couple de fonctions affiche/ masque la position de l'ouvrier qui creuse le labyrinthe en montrant un bloc en matière Or. Le voir travailler d'en haut est amusant et ce mode peut aider à la mise au point si quelque chose cloche.
- **`demolir (realx, realz)`** : détruit un mur de la case désignée par les coordonnées fournies en paramètres d'appel (**`realx, realz`**).
- **`testerMurs (caseX, caseZ)`** : teste si les quatre murs de la case courante sont présents.

> Renvoie la valeur True si c'est le cas ou la valeur False si la case est devenue accessible. Se sert de **`mc.getBlock (x, y, z)`**, qui renvoie le type de matière du bloc désigné. Notez bien les deux signes égalité qui correspondent à l'opérateur de comparaison pour le test qui compare le matériau renvoyé avec celui des murs, qui est stocké dans LABY_MATER.

Nous ajoutons les corps de définition de ces six fonctions en début de programme, juste après la préparation des accès au module Minecraft :

```
def realx(x):
    return LABYX + (x*2) - 1

def realz(z):
    return LABYZ + (z*2) - 1

def montrerBloc(x, z):
    mc.setBlock(realx(x), SOL+1, realz(z), 41)
# 41=Or
def cacherBloc(x, z):
    mc.setBlock(realx(x), SOL+1, realz(z), 0)

def demolir(realx, realz):
    mc.setBlocks(realx, SOL+1, realz, realx,
HAUT+SOL, realz, 0)

def testerMurs(caseX, caseZ):
    if mc.getBlock(realx(caseX)+1, SOL+1,
realz(caseZ))==LABY_MATER and
       mc.getBlock(realx(caseX)-1, SOL+1,
realz(caseZ))==LABY_MATER and
       mc.getBlock(realx(caseX), SOL+1,
realz(caseZ)+1)==LABY_MATER and
       mc.getBlock(realx(caseX), SOL+1,
```

```
realz(caseZ)-1)==LABY_MATER :
        return True

    else:
        return False
```

En cas d'erreur de compilation, commencez par vérifier la présence d'un signe deux-points à la fin des six instructions **`def`**, et des instructions **`else`** et **`if`**.

Le bloc de traitement principal

L'algorithme de construction est répété jusqu'à ce que toutes les cases aient été vues. La condition de répétition est donc tout simplement :

```
while qCasesVues < qCasesTot:
```

La première opération consiste à tester si les cases voisines sont isolées (ont quatre murs) grâce à des appels à **`testerMurs (x, z)`**. Dès qu'une case isolée est trouvée, nous ajoutons sa direction à la liste **possibleDirections[]** au moyen de la méthode prédéfinie **`append()`** que possèdent toutes les listes Python. Cela correspond à l'Étape 3 de l'algorithme. Toutes les lignes qui suivent sont en réalité beaucoup plus décalées par rapport à la marge d'un niveau, car elles sont dans le bloc conditionnel **`while`**.

Si vous êtes dubitatif au niveau de l'indentation des lignes, allez vous repérer dans le listing complet en fin de chapitre.

```
possibleDirections = []

if testerMurs(posX - 1, posZ):
    possibleDirections.append("GAU")

if testerMurs(posX + 1, posZ):
    possibleDirections.append("DRO")

if testerMurs(posX, posZ - 1):
    possibleDirections.append("AVA")
```

```
if testerMurs(posX, posZ + 1):
    possibleDirections.append(“ARR”)
```

Les quatre mots-clés AVA, ARR, GAU et DRO (avant, arrière, gauche, droite) correspondent à des directions dans une surface (même si le jeu est en 3D, cela fonctionne et permet de rester simple). Si vous survolez le dédale en cours de production, ces directions coïncident avec ce que vous voyez si le bloc de départ est situé dans le coin supérieur gauche (LABYX, LABYZ).

Aucun test n’est prévu pour le cas où les tests fassent sortir des limites par un bord. Cela ne pose pas de problème parce qu’une case hors du labyrinthe ne possède pas quatre murs (au mieux, elle a un mur de la limite extérieure). Elle ne peut donc jamais être visitée.

L’étape 4 de notre algorithme consiste à choisir au hasard un des quatre côtés de la case actuellement isolée, de détruire le mur puis d’entrer dans la cellule. Nous testons la longueur de la liste **possibleDirections** qui ne doit pas être vide (expression `! = 0`). Toutes ces instructions sont dépendantes de la boucle de répétition **`while`** vue plus haut.

Juste avant de changer de position, nous cachons le bloc Or de repérage.

```
cacherBloc(posX, posZ)
if len(possibleDirections) != 0:

directionChosen=random.choice(possibleDirections)

    if directionChosen == "GAU":
        demolir(realx(posX) - 1, realz(posZ))
        posX -= 1

    if directionChosen == "DRO":
        demolir(realx(posX) + 1, realz(posZ))
        posX += 1

    if directionChosen == "AVA":
        demolir(realx(posX), realz(posZ) - 1)
        posZ -= 1

    if directionChosen == "ARR":
```

```
        demolir(realx(posX), realz(posZ) + 1)
        posZ += 1
```

Dès que nous sommes dans la nouvelle case, nous augmentons de un le compteur de cases vues et ajoutons la case à la liste qui mémorise le chemin parcouru. Nous montrons à nouveau le bloc Or de repérage pour que la construction soit visible :

```
qCasesVues += 1
listeCasesVues.append((posX, posZ))
montrerBloc(posX, posZ)
```

Le stockage de la liste des cases vues mérite explication. Vous remarquez que les coordonnées **posX** et **posZ** sont placées dans un second jeu de parenthèses, qui désigne le type de données tuple. Un tuple est proche d'une liste, sauf que les valeurs qu'il contient ne sont pas modifiables (le tuple est immuable.) Ainsi, **listeCasesVues** est une liste qui contient des tuples, chaque tuple contenant une paire de valeurs pour X et Z.

Pour afficher le contenu de la liste, il suffit d'utiliser la fenêtre de l'interpréteur Python avec la commande **`print()`**. Voici un exemple tiré d'un essai du jeu ; on voit le parcours dans le labyrinthe :

```
>>> print(listeCasesVues)
[(6, 6), (6, 7), (6, 8), (5, 8), (4, 8), (3, 8),
(3, 7)]
```

Pour l'Etape 5 de l'algorithme, si la case courante n'a pas de voisine non vue, on revient en arrière en supprimant le plus récent couple de coordonnées de la liste avec la méthode de liste **`pop()`**. Nous récupérons ces deux valeurs dans la variable **retrace**, qui contient ensuite un tuple avec le X et le Z de la position actuelle. Comme pour une liste, on peut désigner un contenu par un numéro qui commence à zéro. En écrivant `retrace[0]`, on obtient la valeur X de la position, et `retrace[1]` donne la valeur Z. Voici ces instructions. Nous affichons aussi le bloc Or à la nouvelle position :

```
else:         # Si cases voisines toutes vues
    retrace = listeCasesVues.pop()
    posX = retrace[0]
    posZ = retrace[1]
    montrerBloc(posX, posZ)
```

Le bloc **`else`** doit être aligné comme le bloc **`if`** correspondant, donc à 4 puis 8 espaces de la marge gauche.

La dernière étape de l'algorithme est déjà en place du fait que la boucle de répétition **`while`** va se répéter jusqu'à ce que toutes les cases aient été vues.

Ajouter un toit ?

Nous trouvons plus amusant de laisser le labyrinthe décapoté. On peut ainsi le survoler et choisir de plonger dans un couloir au choix. Mais si vous voulez que les joueurs ne puissent pas tricher, il faut couvrir le dédale. Il faut d'abord, tout au début du programme, remplacer la mention `False` par la mention `True` comme valeur pour la constante **TOIT**. Nous pouvons alors tester cette constante et ajouter une couverture en briques de verre, pour que le dédale ne soit pas noyé dans l'obscurité :

```
if TOIT == True:
    mc.setBlocks(LABYX, SOL+HAUT+1, LABYZ, LABYX+
(DIM*2), SOL+HAUT+1,
LABYZ+(DIM*2), 20)
```

Placer le joueur pour commencer

Il ne reste plus qu'à positionner le joueur dans la case dans laquelle nous avons commencé à produire les murs. Vous pouvez bien sûr le faire démarrer ailleurs, mais cette position n'est pas plus mauvaise qu'une autre et elle permet de réutiliser des valeurs aléatoires déjà disponibles :

```
mc.player.setTilePos(realx(joueurX), SOL+1,
realz(joueurZ))
```

Et c'est tout ! Vous êtes prêt à jouer ! La Figure 13.4 montre l'intérieur du labyrinthe.

Figure 13.4 : Trouver sa voie.

Le code source complet

Le Listing 13.1 déroule la totalité du code source de ce projet.

Listing 13.1 : Programme complet du labyrinthe Minecraft.

```
import sys, random
from mcpi import minecraft
mc = minecraft.Minecraft.create()

mc.postToChat("Bienvenue dans mon MineLabyr !")

def realx(x):
    return LABYX + (x*2) - 1

def realz(z):
    return LABYZ + (z*2) - 1

def montrerBloc(x, z):
    mc.setBlock(realx(x), SOL+1, realz(z), 41) #
41 = Or
```

```
def cacherBloc(x, z):
    mc.setBlock(realx(x), SOL+1, realz(z), 0)

def demolir(realx, realz):
    mc.setBlocks(realx, SOL+1, realz, realx,
HAUT+SOL, realz, 0)

def testerMurs(caseX, caseZ):
    if mc.getBlock(realx(caseX)+1, SOL+1,
realz(caseZ))==LABY_MATER
and mc.getBlock (realx(caseX)-1, SOL+1,
realz(caseZ))==LABY_MATER
and mc.getBlock(realx(caseX), SOL+1,
realz(caseZ)+1)==LABY_MATER and
mc.getBlock(realx(caseX), SOL+1,
realz(caseZ)-1)==LABY_MATER:
       return True
     else:
      return False

mc.setting("world_immutable", True)

# Configuration
DIM = 10
HAUT = 2
LABYX = 0
SOL = 0
LABYZ = 0
LABY_MATER = 1       # 1 = Pierre
SOL_MATER = 2        # 2 = Herbe
TOIT = False

# Nettoyage de la zone
mc.setBlocks(LABYX-10, SOL, LABYZ-10, LABYX+
(DIM*2)+10, SOL+150,
```

```
LABYZ+(DIM*2)+10, 0) # air

# Pose du plancher
mc.setBlocks(LABYX-10, SOL, LABYZ-10, LABYX+
(DIM*2)+10, SOL,
LABYZ+(DIM*2)+10, SOL_MATER)

# Repérage d’origine
mc.setBlock(LABYX, SOL+HAUT+1, LABYZ, LABY_MATER)

# Amener le joueur au dessus du milieu
mc.player.setTilePos(LABYX+DIM, SOL+25,
LABYZ+DIM)

mc.postToChat("Construction du labyrinthe...")

# Construction de la gaufre
for line in range(0, (DIM+1)*2, 2):
    mc.setBlocks(LABYX+line, SOL+1, LABYZ,
LABYX+line, SOL+HAUT,
LABYZ+(DIM*2), LABY_MATER)
    mc.setBlocks(LABYX, SOL+1, LABYZ+line, LABYX+
(DIM*2), SOL+HAUT,
LABYZ+line, LABY_MATER)

# Définition des variables
qCasesTot = DIM * DIM
qCasesVues = 1          # 1 car on commence dans
une case
listeCasesVues = []

posX = random.randint(1, DIM)
posZ = random.randint(1, DIM)
joueurX = posX
joueurZ = posZ
montrerBloc(posX, posZ)
listeCasesVues.append((posX, posZ))
```

```
# BOUCLE PRINCIPALE

while qCasesVues < qCasesTot:
    possibleDirections = []

    if testerMurs(posX - 1, posZ):
        possibleDirections.append(“GAU”)

    if testerMurs(posX + 1, posZ):
        possibleDirections.append(“DRO”)

    if testerMurs(posX, posZ - 1):
        possibleDirections.append(“AVA”)

    if testerMurs(posX, posZ + 1):
        possibleDirections.append("ARR")

    cacherBloc(posX, posZ)

    if len(possibleDirections) != 0:

directionChosen=random.choice(possibleDirections)

        # Abattre un mur dans une direction
choisie
        if directionChosen == "GAU":
          demolir(realx(posX) - 1, realz(posZ))
            posX -= 1

        if directionChosen == "DRO":
            demolir(realx(posX) + 1, realz(posZ))
            posX += 1

        if directionChosen == "AVA":
            demolir(realx(posX), realz(posZ) - 1)
```

```
            posZ -= 1

        if directionChosen == “ARR”:
            demolir(realx(posX), realz(posZ) + 1)
            posZ += 1

        qCasesVues += 1
        listeCasesVues.append((posX, posZ))
        montrerBloc(posX, posZ)

    else:            # Toutes les voisines ont
été vues
        retrace = listeCasesVues.pop()
        posX = retrace[0]
        posZ = retrace[1]
        montrerBloc(posX, posZ)

if TOIT == True:
    mc.setBlocks(LABYX, SOL+HAUT+1, LABYZ, LABYX+
(DIM*2), SOL+HAUT+1,
LABYZ+(DIM*2), 20)

mc.postToChat("Labyrinthe disponible !")
mc.postToChat("Bonne promenade !")
mc.player.setTilePos(realx(joueurX), SOL+1,
realz(joueurZ))
```

Modification du programme

Une fois que le labyrinthe est généré, le bloc en or reste visible. Vous pouvez donc vous donner comme défi de revenir jusqu’à lui. Vous pouvez aussi déposer d’autres blocs spéciaux dans le parcours et chronométrer le temps consommé pour les retrouver tous. La commande **`mc.player.getTilePos()`** renvoie le triplet de coordonnées `x, y, z` de la position actuelle du joueur.

Vous pouvez prévoir une entrée et une sortie en bord de labyrinthe avec une position aléatoire : le joueur doit aller de l’une à l’autre ou créer des

labyrinthes plus grands qui restent jouables en ajoutant des repères dans le paysage ou en haut des murs (ou en variant la matière de certains murs). Pourquoi pas créer un palais des glaces avec des murs en verre ? Et avec un escalier, vous avez un labyrinthe à étages ! Les possibilités sont telles qu'il y a de quoi s'y perdre.

Chapitre 14

De la musique avec Sonic Pi

DANS CE CHAPITRE :

- **Jouer des mélodies avec Sonic Pi**
- **Faire générer de la musique aléatoire**
- **Ajouter des échantillons à une composition**
- **Synchroniser un morceau avec une batterie et des échantillons**

La musique populaire que l'on entend de nos jours, doit beaucoup aux technologies informatiques, et cela ne date pas d'hier. Avec le logiciel Sonic Pi, vous allez pouvoir composer des morceaux en écrivant des programmes sur votre Raspberry Pi. Plusieurs mélodies peuvent être exécutées en même temps avec des rythmiques et des effets, ce qui permet d'obtenir des oeuvres instrumentales tout à fait diffusables en public.

Le créateur de Sonic Pi, Sam Aaron, a précisé : « Sonic Pi a deux objectifs concurrents : il doit être simple à prendre en main, mais aussi assez puissant pour satisfaire le musicien professionnel. Si vous trouvez que cela semble ambitieux, songez que c'est exactement ce que permet un simple piano. »

Sam a donné des concerts Sonic Pi dans les plus grands festivals de musique électronique, comme le Moogfest 2016. Pendant que Sam crée la musique sur scène, les spectateurs peuvent le voir écrire de nouvelles instructions en temps réel. Le magazine *Rolling Stone* a décrit la musique de Sam comme faisant penser à du « Kraftwerk époque *Electric Café*, avec une pointe d'Aphex Twin et de l'électro des années 80 ».

Sonic Pi fonctionne depuis le bureau et il est installé dès le départ avec Raspbian.

Pour démarrer l'application Sonic Pi, ouvrez le menu **Applications**, puis la catégorie **Programmation**. Vous voyez alors l'entrée **Sonic Pi**.

L’interface visuelle de Sonic Pi

La Figure 14.1 donne un aperçu général de l’interface utilisateur de Sonic Pi. Si vous utilisez une version plus récente que nous, vous remarquerez peut-être quelques différences mineures, mais les principes restent les mêmes.

Tout d’abord, agrandissez au maximum la fenêtre pour pouvoir voir le maximum d’éléments. (J’ai expliqué comment retailler les fenêtres dans le Chapitre 4.) En dessous de la barre de boutons, nous voyons dans la partie gauche la zone dans laquelle vous allez saisir les instructions, c’est l’éditeur. Le panneau de droite, intitulé **Trace** (ou Log), est celui dans lequel Sonic Pi affiche les messages pendant l’exécution de votre musique. La partie inférieure de la fenêtre correspond à l’aide en ligne.

Sonic Pi
Run
Stop
Rec
Save
Load
Size
Size
Scope
Info
Help
Prefs
Welcome to Sonic Pi v2.11.1
Buffer 0
Buffer 1
Buffer 2
Buffer 3
Buffer 4
Buffer 5
Buffer 6
Buffer 7
Buffer 8
Buffer 9
Trace
=> Welcome to Sonic Pi v2.11.1
=> Running on Ruby 2.3.1
=> Turn your head towards the sun
and the shadows
will fall
behind you.
Aide
1 Bienvenue dans Sonic Pi
1.1 Codage en 'live'
1.2 Découverte de l'interface
1.3 Apprendre en jouant
2 Synthés
2.1 Vos premiers Beeps
2.2 Paramètres des synthés
Tutoriel
Exemples
synthétiseur
Fx
Échantillons
Langage
Codage en “Live”
Un des aspects les plus passionnants de Sonic Pi est qu'il vous permet d'écrire et de modifier du code en 'live' pour faire de la musique, comme on peut le faire avec une guitare par exemple. Cela veut dire qu'avec un peu de pratique, vous pourrez jouer avec Sonic Pi sur scène et vous amuser avec
Libérez votre créativité
Avant que vous ne plongiez dans les détails réels du fonctionnement de Sonic Pi dans la suite de ce tutoriel, j'aimerais vous faire vivre l'expérience de à quoi ça ressemble de coder en "live". Ne vous inquiétez pas si
Sonic Pi v2.11.1 on Win

Figure 14.1 : Aspect général de la fenêtre de Sonic Pi.

Un point important de Sonic Pi concerne la fenêtre de l’éditeur à gauche : elle comporte 10 pages différentes, correspondant aux 10 onglets **Buffer 0** à **Buffer 9**, dans le bas de la fenêtre. En cliquant un des onglets, vous basculez d’une page de code source à une autre. Dans chaque page peut se trouver un fichier source différent, et vous pouvez lancer l’exécution du contenu de plusieurs pages à la fois. C’est très utile en musique live. Vous pouvez ainsi lancer une boucle de répétition dans une page puis ajouter une mélodie progressivement dans une autre page. Lorsque vous quittez le programme, le contenu des 10 pages est automatiquement enregistré et rechargé au prochain démarrage. Vous

pouvez également choisir d'enregistrer volontairement le contenu d'une page dans un fichier texte au moyen du bouton **Save** de la barre d'outils en haut.

Cette barre d'outils supérieure propose les commandes essentielles :

- **`Run`** pour lancer l'exécution du contenu de la page active ;
- **`Stop`** pour l'arrêter ;
- **`Rec`** pour lancer l'enregistrement ;
- **`Save`** pour sauvegarder ;
- **`Load`** pour recharger un fichier existant.

La partie droite comporte des boutons d'options. Les deux boutons **Size** sont très utiles car celui marqué **Size-** permet de réduire le corps des caractères dans la fenêtre de l'éditeur. Au départ, l'éditeur utilise des caractères vraiment trop grands. N'hésitez pas dès maintenant à cliquer plusieurs fois le bouton **Size-**pour voir au moins la totalité du commentaire initial. Vous disposez également d'un bouton **Scope** qui permet de remplacer l'affichage des messages texte par une visualisation de la forme d'onde que vous entendez. Un bouton **Info** permet d'accéder à des détails au sujet de la fenêtre, un autre bouton **Help** affiche ou masque le panneau inférieur de l'aide et du tutoriel et un dernier bouton **Prefs** sert à modifier les options générales.

Cliquez ce bouton **Prefs** pour accéder à la page des préférences. Vous y trouvez un curseur de réglage du volume et une option pour envoyer la sortie audio vers une prise casque ou vers le câble HDMI. Plusieurs options concernent l'éditeur et la recherche des mises à jour. Si vous comptez utiliser un autre logiciel musical ou un instrument électronique externe, vous pouvez activer les options du groupe **Synthétiseurs et effets** dans les préférences **Audio**.

Vous pouvez régler la largeur et la hauteur relative des panneaux en amenant le pointeur de souris sur une frontière. Dès que le pointeur change d'aspect, cliquez pour déplacer celle-ci.

Lorsque vous utiliserez Sonic Pi sur scène, vous tirerez profit du réglage de couleur d'affichage **`Mode sombre`** (préférences **`Editeur`**) qui sera plus approprié au niveau de lumière d'un club ou d'une salle de concert.

Vos premières notes

Cliquez dans la zone de l'éditeur en haut à gauche et saisissez l'instruction suivante :

```
play 60
```

Lorsque vous validez la saisie avec la touche Entrée, il ne se passe rien. Pour faire exécuter cette instruction, il faut utiliser le bouton **Run**. Vous allez entendre un do moyen, appelé do du milieu du clavier. Un message apparaît dans la fenêtre de trace à droite.

Lorsque l'on utilise des nombres pour les notes, ce sont les nombres de la norme MIDI qui est extrêmement répandue dans le monde de la musique électronique. Nous les avons déjà rencontrés dans Scratch. Plus la valeur est élevée, plus la note est aiguë. D'une valeur à la suivante, la différence est d'un demi-ton.

Ajoutons deux notes, situées quatre et sept demi-tons au-dessus de la première :

```
play 60
play 64
play 67
```

Lorsque vous testez ce programme avec le bouton **Run**, vous entendez dorénavant trois sons en même temps. Il s'agit tout simplement d'un accord de do majeur qui est constitué d'un do (valeur 60), d'un mi (valeur 64) et d'un sol (valeur 67). Pour ne pas jouer les notes simultanément, mais successivement, comme dans un arpège, il suffit d'ajouter une instruction de pause entre deux notes au moyen de la commande **`sleep`** :

```
play  60
sleep 0.5
play  64
sleep 0.5
play  67
sleep 0.5
play  72
```

Dans cet exemple, j'ai ajouté une autre note do deux fois plus aiguë que la première (valeur 27), ce qui donne à l'exemple un petit air de fanfare. En quelques lignes, vous en savez déjà assez pour composer vos propres mélodies. Il s'agit de bien choisir vos séquences de notes.

Le Tableau 14.1 donne la correspondance entre les noms des notes musicales et les valeurs des notes du standard MIDI, de 0 à 127. En pratique, la colonne 0 et les deux colonnes 8 et 9 génèrent des sons soit trop graves, soit vraiment trop aigus pour donner un résultat acceptable, cela dépendant du son choisi. En règle générale, il est conseillé de n'utiliser que les valeurs de notes entre 48 et 96, mais n'hésitez pas à faire des essais.

Tableau 14.1 : Équivalence entre notes et valeurs MIDI.

Note		0	1	2	3	4	5	6	7	8	9
C	0	12	24	36	48	60	72	84	96	108	120
C#	1	13	25	37	49	61	73	85	97	109	121
D	2	14	26	38	50	62	74	86	98	110	122
D#	3	15	27	39	51	63	75	87	99	111	123
E	4	16	28	40	52	64	76	88	100	112	124
F	5	17	29	41	53	65	77	89	101	113	125
F#	6	18	30	42	54	66	78	90	102	114	126
G	7	19	31	43	55	67	79	91	103	115	127
G#	8	20	32	44	56	68	80	92	104	116	
A	9	21	33	45	57	69	81	93	105	117	
A#	10	22	34	46	58	70	82	94	106	118	
B	11	23	35	47	59	71	83	95	107	119	

Dans ce tableau, chaque octave correspond à une colonne. La lecture se fait donc dans le sens vertical entre le do (lettre C), et le si (lettre B).

Vous remarquez que nous utilisons les noms anglo-saxons pour les notes. En guise de rappel, voici l'équivalence :

C = do, D =ré, E = mi, F = fa, G = sol, A = la, B = si.

Si vous n’avez aucune connaissance en solfège, ce n’est pas grave. Pour vos premiers pas, n’utilisez aucune note altérée, c’est-à-dire dont le nom est suivi d’un signe dièse. Évitez également les sauts trop importants entre deux notes. Dans vos mélodies, progressez de quelques notes vers le haut ou vers le bas et changez de colonne dans le tableau lorsque vous êtes non loin du bas d’une colonne pour aller à la suivante ou du haut d’une colonne pour revenir à la précédente. En appliquant ces quelques règles de bon sens, vous obtiendrez toujours des résultats assez mélodieux.

Noms des notes et accords

Dans Sonic Pi, vous n’êtes pas forcé d’utiliser les valeurs numériques pour les notes. Vous pouvez également indiquer directement la lettre anglo-saxonne de A à G, mais dans ce cas il faut faire suivre cette lettre du numéro de l’octave. Ce numéro correspond à l’en-tête de colonne tout au début du Tableau 14.1.

Pour jouer par exemple le même do du milieu du clavier que nous avons déjà essayé (valeur 60), nous pouvons écrire :

```
play  :c4
```

Pour jouer le si qui se trouve juste à gauche du do sur un clavier, et qui est donc dans la colonne précédente tout en bas dans le Tableau 14.1, nous écrivons :

```
play  :b3
```

Le panneau des messages Trace confirme que Sonic Pi a joué successivement les notes 60 puis 59. En regardant le Tableau 14.1, vous pouvez vérifier que c’est ce que vous vouliez faire.

Voici comment réécrire notre petite fanfare avec des noms de notes :

```
play   :c4
sleep 0.5
play   :e4
sleep 0.5
play   :g4
sleep 0.5
```

```
play   :c5
```

Pour désigner une note augmentée d'un demi-ton par le signe #, il faut ajouter la lettre **s** à la suite du nom de la note, comme dans . Pour diminuer une note d'un demi-ton, vous utilisez la lettre **b** (pour bémol), comme dans .

Vous pouvez même faire jouer des accords complets sans devoir indiquer les noms de chacune des notes de ces accords. Il suffit d'utiliser la fonction **chord()** avec un des noms prédéfinis dans Sonic Pi. Vous indiquez d'abord le nom de la fondamentale, c'est-à-dire la note de base de l'accord, et vous ajoutez le nom du type d'accord. Les plus utilisés sont **: major**, **: minor**, **: diminished**, **: major7**, **: minor7 :** , **: diminished7** et **: dom7**, mais il y en a d'autres. Pour la liste complète, ouvrez le panneau d'aide, choisissez la catégorie Langage à gauche puis la fonction **chord**. Voici un essai d'exécution de deux accords :

```
play chord(:a3, :major)
sleep 1
play chord(:a3, :minor)
```

Dans les deux instructions, trois notes sont émises en même temps. Si vous observez le contenu de la fenêtre des messages, vous pouvez constater que la note du milieu était un demi-ton plus bas dans le deuxième accord parce que c'est un accord mineur. Vous pouvez vérifier à quelles notes correspondent ces accords en vous servant du Tableau 14.1.

La fonction **chord** renvoie les noms des notes de l'accord sous forme d'une liste. Vous pouvez donc utiliser la fonction **play_pattern()** pour faire jouer les notes de l'accord l'une après l'autre, sous forme d'arpège, comme dans ces exemples :

```
play_pattern chord(:a3, :major)
play_pattern chord(:a3, :minor)
```

Pour alléger l'écriture

L'instruction **play_pattern_timed()** permet de définir en une fois les notes d'une mélodie et les durées entre deux notes. Dans le panneau de

l'éditeur, cliquez dans un bouton d'onglet menant à une page encore vide et saisissez l'instruction suivante :

```
play_pattern_timed [:c4, :e4, :g4, :c5], [0.5,
0.5, 1]
```

Soyez attentif aux crochets droits et aux signes deux-points. Cette instruction travaille avec deux listes, chacune étant délimitée par un jeu de crochets. La première liste indique les notes à jouer, qui sont les mêmes que dans notre exemple précédent. La seconde liste est séparée de la précédente par une virgule. Elle indique les durées des pauses entre notes. Il y a donc quatre notes et trois pauses. Nous avons ici choisi une pause d'un demi-temps entre les première et deuxième puis entre les deuxième et troisième notes, et une pause d'un temps complet avant la dernière note. Cela ajoute un peu de suspens avant cette dernière note.

Une mélodie aléatoire avec Shuffle

Tout ce qui est écrit entre deux crochets droits est une liste, du même genre que les listes de Python. Une liste Python est dotée automatiquement de plusieurs fonctions appelées méthodes qui permettent de jouer sur la manière dont le contenu est utilisé. Voici comment utiliser la méthode **`reverse`** d'une liste :

```
play_pattern_timed [:c4, :e4, :g4, :c5], [0.5,
0.5, 1]
play_pattern_timed [:c4, :e4, :g4, :c5].reverse,
[0.5, 0.5, 1]
```

Les deux instructions précédentes font d'abord jouer les quatre notes dans le sens montant, puis dans le sens descendant. Si nous utilisons la méthode **`shuffle`**, nous modifions l'ordre des éléments de la liste, ce qui donne une mélodie errante. Essayez ceci (tout sur la même ligne) :

```
play_pattern_timed [:c4,  :d4,  :e4,  :f4,  :g4,
:a4,  :c5].shuffle, [0.5, 0.5, ↲
1, 0.5, 0.5, 1]
```

Le rythme choisi est vraiment très simple : deux brèves et une longue (comme dans *Vive le vent*). La mélodie n'est pas désagréable, mais elle est un peu courte. Nous pouvons demander à Sonic Pi de la répéter :

```
4.times do
  play_pattern_timed [:c4,  :d4,  :e4,  :f4,
:g4,  :a4,  :c5].shuffle, [0.5, ↲
0.5, 1, 0.5, 0.5, 1, 2]
end
play  :c4
```

Nous avons ajouté une boucle de répétition avec **`.times`**, ce qui va faire répéter quatre fois l'instruction. Le début de la boucle correspond à **`4.times do`**. La fin du bloc à répéter correspond au mot **`end`**. Entre les deux, Sonic Pi a automatiquement ajouté deux espaces en marge gauche pour confirmer que l'instruction **`play_pattern_timed()`** fait partie de la boucle à répéter. Il suffit bien sûr de changer la valeur 4 au début pour changer le nombre de répétitions.

Vous remarquez en outre que nous avons ajouté une valeur de durée pour la dernière note, la valeur 2. Nous avons enfin prévu de faire jouer une dernière note en dehors de la boucle, un do, c4. Ainsi, quelle que soit la qualité la mélodie obtenue par tirage au sort, la séquence se terminera toujours correctement sur un do, ce qui est logique puisque toutes les notes que nous demandons de jouer font partie de la gamme de do majeur.

Au gré du hasard

Si vous faites jouer plusieurs fois ce petit exemple, vous constatez que ce sont toujours les mêmes notes qui sont choisies par le hasard. Cela a été prévu ainsi par le concepteur de Sonic Pi : lorsque vous exécutez plusieurs fois un morceau, même s'il comporte des éléments aléatoires, le résultat sonore sera toujours le même. Vous gardez ainsi le contrôle en tant que compositeur et vous saurez ce que les auditeurs vont entendre, même si vous incorporez du hasard.

Dans certains cas, vous aurez envie de modifier la sélection aléatoire. Pour ce faire, il suffit de changer le germe du générateur de valeurs aléatoires. Par défaut, la valeur est égale à zéro. Pour générer des mélodies qui changent d'une exécution à l'autre, il suffit d'ajouter la ligne suivante tout au début du programme :

```
use_random_seed 10
```

Si vous changez la valeur du germe, vous obtenez une autre variation dans la sélection aléatoire.

Nommer vos listes

Lorsque vos listes de routeur de son, c'est-à-dire vos notes, et vos listes de durée commencent à devenir longues, il devient confortable de les répéter dans vos programmes. Dans ce cas, n'hésitez pas à donner un nom à chacune de vos listes. Vous pouvez ensuite vous servir directement des noms. Voici une reformulation du programme précédent grâce à deux listes nommées :

```
use_random_seed 10
hauteur = [:c4,  :d4,  :e4,  :f4,  :g4,  :a4,
:c5]
duree   = [0.5, 0.5, 1, 0.5, 0.5, 1, 2]
4.times do
  play_pattern_timed hauteur.shuffle, duree
end
play 60
```

Faire jouer des hauteurs au hasard

Voici comment vous pouvez faire jouer une séquence de notes sélectionnées de façon aléatoire :

```
plusBasse = 60
plusHaute = 84
6.times do
  play rrand_i(plusBasse, plusHaute)
  sleep 0.5
end
```

Vous saisirez cet exemple dans une nouvelle page de l'éditeur. Les deux premières lignes déclarent deux variables. La fonction **`rrand_i()`** renvoie une valeur entière choisie au hasard. Nous indiquons dans ces parenthèses les deux valeurs minimale et maximale que la fonction peut sélectionner. Le fonctionnement est proche de celui de la méthode **`shuffle()`** : c'est toujours la même séquence qui est choisie au hasard. Nous avons vu qu'il suffisait de changer le germe lorsque vous ne vouliez pas que ce soit toujours ainsi.

Le problème de la génération de valeurs aléatoires est que toutes les notes ne résonnent pas correctement ensemble. Dans le premier exemple du chapitre, nous n'avions choisi que les notes de la gamme de do majeur, et aucune note avec un dièse. Si vous commencez à laisser introduire des notes altérées dans la mélodie (ce que permet le choix des deux limites 60 et 84), le résultat va commencer à devenir désagréable. Pour éviter cela, il suffit de faire choisir au hasard dans une liste de notes prédéfinies, par exemple celles de la gamme de do majeur, en utilisant la méthode **`choose()`**. Saisissez l'exemple dans une autre page de l'éditeur :

```
hauteur = [:c4,  :d4,  :e4,  :f4,  :g4,  :a4,
:c5]
loop do
  play hauteur.choose()
  sleep 0.5
end
```

Cette boucle est éternelle. La mélodie improvisée ne s'arrête que si vous utilisez le bouton **Stop**.

Vous pouvez voir les valeurs des notes qui ont été choisies dans la fenêtre de trace, ce qui permet de confirmer que seules les notes de la liste fournie sont utilisées.

Le temps réel avec live_loop

Une des fonctions les plus puissantes de Sonic Pi est la boucle live ou boucle continue qui correspond au mot-clé **`live_loop`**. Elle permet de modifier le contenu de votre programme sans en arrêter l'exécution.

Restez dans la page du programme précédent et modifiez-le comme le propose le listing suivant. Nous transformons la boucle éternelle en boucle live et nous ajoutons des indications de tempo (BPM signifie *beats per minute*), le nom d'un synthétiseur et quelques options. Le fait de changer de synthétiseur revient à changer le son avec lequel seront exécutées les notes. Les options peuvent être ajoutées après l'instruction **`play`** qui demande de jouer une note. Elle permet de contrôler la façon dont le son va être émis avec l'instrument choisi. Par exemple, l'option intitulée **`attack`** que nous utilisons détermine le temps nécessaire pour que la note atteigne son volume initial maximal, ce qui permet de jouer sur le côté percussif du timbre.

Cet exemple complet reprend la boucle définie dans la section précédente. Vous pouvez le charger depuis le dossier des exemples. Il porte le nom *flotcontinu.txt*.

Listing 14.1 : flotcontinu.txt.

```
pentamaj = [:c4,  :d4,  :e4,  :g4,  :a4,  :c5]
live_loop :flotcontinu do
  use_bpm 120
  use_synth :dull_bell
  play pentamaj.choose(), attack: 0.01
  sleep 0.5
end
```

Dans Sonic Pi, il faut prendre l'habitude de toujours rester attentif à la position des signes deux-points. Parfois, le signe deux-points est en préfixe, collé au début d'un mot, et parfois en suffixe, collé à la fin.

- Le signe deux-points est en préfixe pour les noms des notes, des accords, des instruments, des boucles et des échantillons, comme dans **`:dull_bell`**. Si vous insérez une espace, l'objet n'est plus reconnu.
- Le signe deux-points est en suffixe pour les noms des options et paramètres dont vous pouvez décider de la valeur. C'est le cas dans l'exemple d'**`attack :`** suivi d'une espace et de la valeur désirée.

Quand vous définissez une boucle live, vous devez lui donner un nom unique. Dans l'exemple, ce nom est **`:flotcontinu`**. N'utilisez cependant aucune lettre accentuée, ni caractère spécial.

Si vous lancez l'exécution de l'exemple, vous allez entendre le nouveau son de synthétiseur avec le tempo demandé (**`:dull_bell`**).

Un des atouts majeurs de Sonic Pi est que vous pouvez facilement intervenir sur votre programme, et même s'il est en cours d'exécution. Par exemple, sélectionnez le nom du synthé dans l'éditeur et saisissez **`:`**

`chipbass` puis cliquez **Run**. La mélodie se poursuit avec le nouveau son, sans perte du tempo. Réduisez le tempo avec **`use_bpm`** à 60 et cliquez **Run** : tout se ralentit. Augmentez le paramètre **`attack :`** jusqu'à 1 : le son devient mou, comme une sorte de bourdonnement d'orgue. Vous pouvez ajouter des options en les séparant par des virgules.

Les retouches que vous réalisez dans le code source ne sont prises en compte par Sonic Pi qu'à partir du moment où vous cliquez **Run**.

Le panneau **Aide** propose une description de plusieurs dizaines de synthétiseurs préinstallés. Affichez le panneau **Aide** puis sélectionnez la catégorie **Synthés** en bas. Sélectionnez un synthé à gauche puis faites défiler le volet droit qui décrit tous les paramètres applicables.

Vous pouvez même changer de synthé pendant l'exécution en tirant profit du mécanisme d'aide à la saisie. Effacez d'abord le nom actuel, en laissant le signe deux-points, puis saisissez la première lettre du son désiré. Vous voyez s'afficher la liste des synthés commençant par cette lettre (Figure 14.2.) Sélectionnez celui désiré au curseur ou par clic puis confirmez par Entrée. Cette technique permet d'explorer facilement tous les sons de synthèse disponibles.

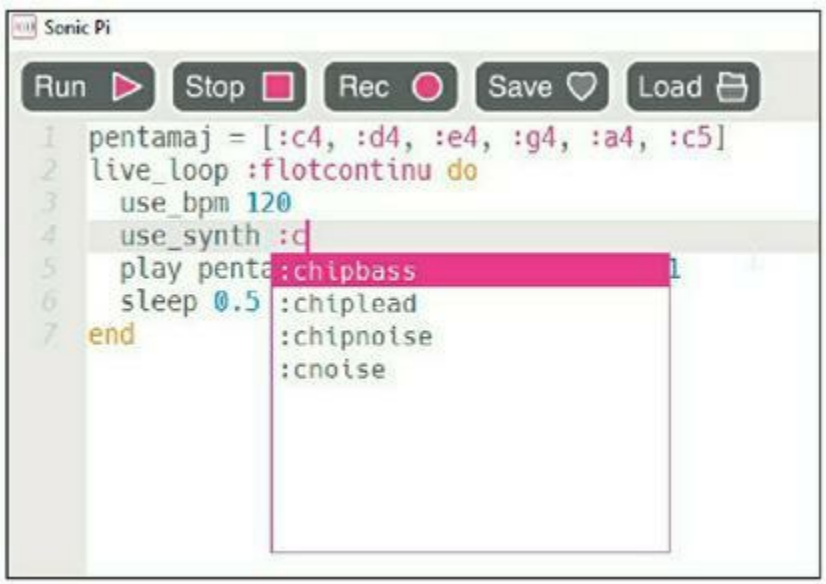


Figure 14.2 : Sélection d'un synthé avec l'aide à la saisie.

Si vous cliquez plusieurs fois **Run** pour une partie qui ne définit pas une boucle live, vous faites démarrer plusieurs exécutions décalées, parce que non synchronisées.

Reproduire des échantillons

Les morceaux créés depuis le début du chapitre forment déjà une excellente base pour produire de la musique électronique, d'autant que vous pouvez laisser une part de hasard venir vous surprendre. Mais tous les sons entendus jusqu'ici sont des sons générés par le processeur.

Sonic Pi est aussi capable de lire des échantillons (*samples*), c'est-à-dire des bribes audio qui ont été capturées dans le monde physique et que vous pouvez traiter, déformer, ralentir et accélérer. La liste des sons échantillonnés livrés avec Sonic Pi est disponible dans le panneau **Aide** par la catégorie **Echantillons**. Les échantillons sont notamment le format sonore privilégié pour les sons percussifs des batteries et autres rythmiques.

Voici un de nos échantillons favoris :

```
sample :loop_industrial
```

Ouvrez une page d'édition (*buffer*) vide et insérez-y cette instruction puis cliquez **Run**. Avec le paramètre **`rate :`** , vous pouvez par exemple le faire jouer deux fois moins vite :

```
sample :loop_industrial, rate: 0.5
```

Pour obtenir un rythme ou une nappe, il suffit de faire répéter cet extrait. Comme pour les notes MIDI, nous utilisons la commande de silence **`sleep`** pour que les répétitions démarrent au bon moment. Le problème est que les échantillons ont quasiment tous une durée différente. Il faut de ce fait tâtonner pour trouver la bonne durée de silence à spécifier pour **`sleep`** lorsque deux échantillons doivent redémarrer en même temps. Heureusement, Sonic Pi est doté d'un module de compression/étirement temporel (*time stretch*) pour que l'échantillon dure un nombre de temps demandé. Voici comment utiliser cette option **`beat_stretch :`** :

```
loop do
  sample :loop_industrial, beat_stretch: 2
  sleep 2
end
```

Avec cette option, l'échantillon dure exactement deux temps et nous obligeons à marquer deux temps de silence avant de répéter. Le résultat est un rythme ininterrompu. Comme pour les autres sons, vous contrôlez la vitesse d'exécution avec le paramètre **`use_bpm`**.

Ajouter des effets spéciaux (Fx)

Lorsque vous faites jouer un échantillon, vous pouvez lui faire subir des traitements appelés effets spéciaux, les plus connus étant la réverbération, l'écho et la distorsion. Sonic Pi est doté d'une gamme complète d'effets numériques ; ils sont décrits dans la catégorie **Fx** du panneau **Aide**. Comme pour les sons de synthés, vous pouvez profiter de l'aide à la saisie du nom d'effet dans l'éditeur. Voici comment appliquer de la distorsion à un échantillon de guitare :

```
with_fx :distortion do
  sample  :guit_e_fifths
end
```

Plusieurs options sont possibles pour mieux contrôler l'effet. Pour l'effet de distorsion, l'option **`distort :`** contrôle la proportion du signal qui doit être distordu et l'option **`mix :`** détermine la proportion en sortie entre le signal inchangé et celui traité. Pour les deux options, la valeur doit être située entre 0 et 1. Voici un exemple d'utilisation :

```
with_fx :distortion, distort: 0.9, mix: 0.5 do
  sample  :guit_e_fifths
end
```

Faites plusieurs essais en changeant les valeurs puis en relançant par **Run** (ou Alt + R). Testez aussi les autres effets (**`: krush`**, **`: wobble`**, *etc*.).

Pour voir la liste de toutes les options d'un effet, dans le panneau **`Aide`**, cliquez **`Fx`** pour accéder à cette catégorie. Dans la fenêtre de l'éditeur, cliquez dans le nom de l'effet puis frappez les touches Ctrl + I pour afficher la description si elle existe. Ce raccourci d'information Ctrl + I s'applique aussi aux instructions et aux commandes.

Synchronisation avec une piste rythmique

Nous savons maintenant comment répéter un son pour en faire un rythme, et comment faire jouer en même temps une mélodie avec un son de synthé. Plusieurs boucles live peuvent être démarrées en même temps.

Un souci qui doit absolument être réglé est le maintien de la synchronisation entre plusieurs échantillons dont les durées sont différentes. Sonic Pi comporte une option qui permet de rendre le démarrage d'une boucle continue dépendant d'une autre boucle continue. Cette option, qui s'écrit **sync :** , doit être insérée avant le **do**, après une virgule à la suite du nom de la boucle esclave et suivie du nom de la boucle maître (**:batri**), comme dans cet exemple :

```
live_loop :cymbales, sync:  :batri do
```

Le morceau suivant synchronise une boucle de cymbale avec une boucle de batterie :

Listing 14.2 : synchrobatri.txt.

```
live_loop :batri do
  sample :loop_industrial, beat_stretch: 2
  sleep 2
end

live_loop :cymbales, sync:  :batri do
  4.times do
    sample :elec_cymbal
    sleep 1
  end
  sleep 8
end
```

Quand vous lancez cet extrait, vous entendez d'abord quatre coups de cymbale, parfaitement synchronisés avec la batterie, puis huit temps de silence avant que la boucle de cymbales reprenne au début.

Soyez vigilant avec les signes deux-points de l'option **sync :** . Il y a d'abord un signe deux-points accolé en suffixe du nom d'option puis un autre en préfixe du nom de la boucle live avec laquelle ce son doit être synchronisé (**:batri**).

Récapitulons

Le Listing 14.2 propose une synthèse des techniques abordées dans ce chapitre. Il combine cinq boucles continues :

- La boucle fondamentale est celle de la rythmique, **: batri**. Les quatre autres boucles lui sont liées par **sync :** . Cette rythmique exploite l'échantillon `loop_industrial`.
- S'y ajoute une boucle de **: melodie :** (sans accent) qui enchaîne au hasard les notes d'une liste. La même note est présente plusieurs fois dans la liste, ce qui renforce le sentiment de tonalité de la gamme de do (c3 et c4).
- Une autre boucle, **: plinks**, saupoudre la mélodie de quelques notes de la gamme pentatonique de do.
- Le tout est soutenu par une ligne de **: basse** exécutée par une instruction **`play_pattern_timed`**.
- Pour couronner le tout, nous ajoutons après 16 temps de silence la boucle rythmique **: amen** fondée sur un échantillon de batterie classique nommé **: `loop_amen`** que nous répétons huit fois.

Listing 14.3 : Une œuvre complète (robotjam).

```
# Robot jam par Sean McManus
# VF par O. Engler

live_loop :batri do
  with_fx :krush do
    sample :loop_industrial, beat_stretch: 2
  end
```

```
  sleep 2
end

live_loop :melodie, sync:  :batri do
  hauteur = [:c3,  :c3,  :c3,  :d3,  :g3,  :g3,
:a3,  :c4,  :c4]
  use_synth :saw
  with_fx :wobble do
    play hauteur.choose()
    sleep 2
  end
end

live_loop :plinks, sync:  :batri do
  pentaplink = [:c4,  :d4,  :e4,  :g4,  :a4]
  use_synth :chiplead
  play pentaplink.choose()
  sleep 0.25
end

live_loop :basse, sync:  :batri do
  use_synth :tb303
  play_pattern_timed [:c2,  :c2,  :c2,  :c2,
:c2,  :c2,  :g2,  :a2], [0.5, 0.5,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5]
end

live_loop :amen, sync:  :batri do
  sleep 16
  8.times do
    sample :loop_amen, beat_stretch: 2
    sleep 2
  end
end
```

Vous pouvez vous servir de cette « partition » comme point de départ de vos créations. Comme elle est fondée sur des boucles continues, vous

pouvez éditer tous les paramètres (synthés, échantillons, effets) en temps réel, pendant que le morceau s'exécute. Il suffit de cliquer le bouton **Run** pour faire prendre en compte vos retouches.

Plus loin avec Sonic Pi

Ce chapitre vous a sans doute donné envie de plonger dans la création sonore par algorithme sur votre Raspberry Pi ? Vous savez faire jouer des mélodies en choisissant des synthés et des échantillons. Vous avez appris comment faire générer des ambiances sonores en ajoutant du hasard à partir de valeurs de notes ou d'une liste. Vous avez enfin vu comment maintenir plusieurs événements sonores en rythme grâce à la synchronisation. Les exemples du chapitre et le grand nombre d'instruments virtuels fournis dès le départ vous assurent de pouvoir composer avec Sonic Pi des créations inouïes.

Sonic Pi permet aussi de s'ouvrir à d'autres mondes, par exemple celui de Minecraft, et d'exploiter des signaux MIDI pour piloter des instruments électroniques et des logiciels compatibles MIDI. Le panneau **Aide** comporte une importante catégorie **Tutoriel** qu'il est conseillé d'étudier à fond.

Que vous soyez passionné de hip-hop, de dance, de pop ou de rock, Sonic Pi est peut-être le musicien qu'il manque à votre groupe !

PARTIE 5
Découvrir l’électronique avec le Raspberry Pi

DANS CETTE PARTIE :

- Découverte des principes de l'électricité, du calcul d'un courant et des techniques de soudure.
- Accès au monde extérieur au Raspberry Pi grâce à son connecteur GPIO.
- Contrôle des broches d'entrée du Raspberry Pi en langage Scratch et Python.
- Création d'un dé électronique et d'un simulateur de passage piétons.
- Jonglage avec des LED multicolores.
- Création d'un jeu *Rainbow Invaders*.
- Maîtriser les communications RFID.

Chapitre 15

Principes des circuits électroniques et soudure

DANS CE CHAPITRE :

- **Principe d'un circuit électronique**
- **Découverte du connecteur GPIO**
- **Premiers pas avec un fer à souder**
- **Cartes d'extension du commerce**

Cette partie du livre vous propose de plonger dans l'informatique physique, c'est-à-dire les techniques permettant de créer un pont entre l'ordinateur et le monde réel, au-delà du simple trio de base clavier/souris/écran. Vous allez apprendre comment profiter de ce que vous avez appris en programmation Python pour détecter ce qui se passe et pour contrôler des lumières, des moteurs et tout autre objet fonctionnant à l'électricité. Mais pour pouvoir y parvenir en toute sécurité, sans risque de dégâts à votre personne ou à votre circuit Raspberry, il vous sera utile de connaître ou de revoir les grands principes de l'électricité et de l'électronique.

Nous allons donc passer en revue les concepts essentiels qui vous permettront de comprendre pourquoi les projets que nous allons construire se présentent sous une certaine forme et ce qu'il faut éviter de faire. Nous découvrirons le concept du connecteur d'entrées/sorties GPIO, en expliquant ce qu'est une connexion et en indiquant à quoi servent ces connexions avec le Raspberry Pi.

Le premier projet du Chapitre 16 peut être réalisé sans aucune soudure, mais pour pouvoir bien progresser en électronique, il est bon d'apprendre à manier un fer. Nous verrons comment y parvenir en toute sécurité. Enfin, même si tous les projets du livre peuvent être réalisés sans acquérir une carte d'extension, nous décrirons rapidement quelques cartes disponibles dans le commerce qui peuvent simplifier vos travaux.

Qu’est-ce qu’un circuit électrique ?

Il faut d’abord savoir qu’un circuit électrique est quelque chose qui permet l’écoulement de l’électricité, comme un chemin ou un tube. Ce chemin doit être ininterrompu et former une boucle. S’il y a une impasse, ce n’est pas un circuit. L’électricité doit pouvoir s’écouler d’une origine à une destination. Soyons un peu plus précis au sujet du terme *électricité*. C’est en effet un sujet qui peut devenir complexe et la chose dont nous parlons est invisible. Il va donc falloir faire preuve d’un peu d’imagination pour comprendre ce qui se passe. Nous nous intéresserons à deux aspects de l’électricité : le courant et la tension (le voltage).

Le *courant* désigne la quantité d’énergie qui circule. La *tension* est la force de l’écoulement du courant dans le circuit. Une tension ne peut pas s’écouler et il n’y a pas de courant s’il n’y a pas de tension. En revanche, il peut y avoir une tension sans courant. Vous avez certainement vécu les effets de l’électricité statique qui s’accumulent lorsqu’un isolant (un matériel qui ne conduit pas l’électricité) est frotté contre un autre. Si vous frottez un ballon de baudruche avec de la laine, vous pouvez ensuite l’approcher de vos cheveux pour les faire se dresser sur la tête. Vous sentez l’effet, mais seulement parce que vos cheveux se dressent. Vous ne sentez pas l’électricité. Pour sentir l’électricité statique, il faut qu’un courant s’écoule, auquel cas elle n’est plus statique. Notez qu’un très faible courant peut faire mal si la tension est très forte. Vous avez sans doute déjà ressenti une décharge en touchant une poignée de porte après avoir marché sur une moquette en nylon.

Nature de l’électricité

Alors, qu’est-ce que le courant électrique ? Ce sont des électrons qui se déplacent d’atome en atome, comme des voitures à un feu vert. Dans un circuit électrique, le courant se mesure en *ampères*. 1 ampère de courant correspond à 6.24×10^18 électrons par seconde, c’est-à-dire 624 suivi de 16 zéros. C’est un grand nombre, mais nous n’aurons pas à nous soucier de tous ces zéros. Sachez que plus la tension est élevée, plus la quantité de courant qui circule est importante, sauf que tout circuit possède une propriété qui freine l’écoulement, et c’est la *résistance*.

Elle dépend de la matière utilisée pour réaliser le circuit et se mesure dans une unité appelée *ohm*. Nous savons définir un ampère en terme de quantité d’électrons qui circule, et nous pouvons donc définir les deux autres propriétés par rapport à un ampère.

Un volt est la tension nécessaire pour faire s'écouler un ampère dans un circuit dont la résistance est de un ohm.

Cette simple formule permet d'aller très loin en électronique. D'ailleurs, elle correspond à une formule qui s'appelle la loi d'Ohm :

```
Volts = Amps × Ohms
```

Mais cette formule devait paraître trop facile à comprendre. Après tout, il est toujours tentant de se cacher derrière un jargon pour rester entre soi. Imaginez l'impression que vous donnerait votre médecin s'il vous expliquait de façon intelligible la nature de votre problème ! Non, il faut rendre les choses un peu obscures pour que tout le monde ne comprenne pas. Et voilà ce qu'est devenue la loi d'Ohm :

```
E = I × R
```

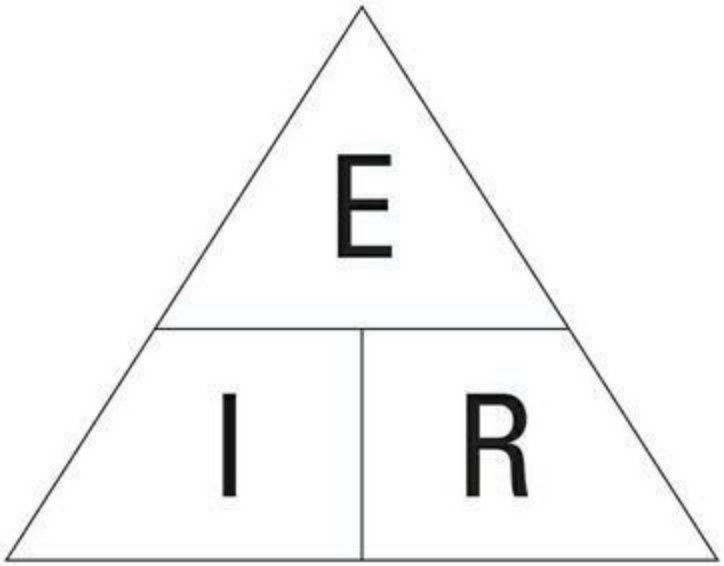
En fait, E correspond à la force électromotrice qui se mesure en volts, I correspond au courant en Ampères et R à la résistance en ohms.

C'est cette formule que vous voyez partout dans les livres et sur Internet. Mais n'oubliez pas que c'est exactement pareil que ceci :

```
tension = courant × résistance
```

Pour relier le Raspberry Pi au monde extérieur, nous allons jongler avec des tensions et des courants, et il nous faudra souvent ajouter une résistance pour limiter le courant qu'une tension donnée va vouloir faire circuler dans un circuit. La loi d'Ohm nous suffira à trouver la bonne valeur. Nous verrons dans le Chapitre 17 comment nous en resservir pour que des diodes lumineuses LED brillent ni trop, ni trop peu.

Nous pouvons calculer d'autres valeurs que la résistance. Dès que nous connaissons deux des trois quantités dans un circuit, nous pouvons trouver la troisième. Il suffit de réarranger la formule en conséquence. Je trouve le triangle de la loi d'Ohm très suggestif, car il permet de retrouver les trois formules d'un seul coup d'oeil.



$E = I \times R$
$I = E / R$
$R = E / I$

À l'époque où les scientifiques ont découvert l'électricité, ils ont tout de suite compris qu'elle circulait d'une borne à une autre et ont décidé que les électrons circulaient de la borne positive vers la borne négative. Mais comment les distinguer ? Ils avaient fait des expériences consistant à faire circuler un courant dans une solution d'eau et de sulfate de cuivre et ont pu constater que le cuivre était dissous d'un côté puis redéposé de l'autre. Ils ont donc logiquement supposé que le métal était transporté dans le même sens que l'électricité. Voilà pourquoi ils ont donné au côté d'où le cuivre partait le nom *anode* (*positif*) et au côté qui recevait le métal le nom *cathode* (le côté *négatif*). Une seule erreur était possible : ils l'ont faite. Les électrons qui incarnent le courant circulent en réalité en sens inverse. Mais la convention d'origine s'est tellement répandue qu'elle continue à être utilisée de nos jours. Pour bien la reconnaître, nous l'appelons le *sens conventionnel du courant*, qui va du plus vers le moins.

En général, le sens, réel ou conventionnel, a moins d'importance que le fait que du courant circule. Nous avons besoin des deux termes positifs et négatifs pour savoir entre quoi et quoi ce courant circule. Les sources d'énergie telles que les piles électriques et les alimentations comportent toutes un marquage du positif et du négatif pour que vous puissiez les connecter correctement. Ce sont toutes des sources de courant continu CC (*Direct Current*, DC en anglais) parce que le courant circule logiquement dans un seul sens.

Mais il existe également des sources d'énergie dans lesquelles le courant change de sens fréquemment : ce sont les sources de courant alternatif (AC). C'est le cas du réseau EDF à 50 cycles par seconde. Les

enseignants en électricité aiment beaucoup jouer à leurs élèves débutants le tour consistant à leur demander d'aller leur acheter une batterie à courant alternatif.

Pour interrompre ou établir un circuit, on utilise un interrupteur.

De la théorie à la pratique

Pour voir comment cela fonctionne, étudions le circuit le plus simple qui soit. Pour que les choses soient claires, nous utilisons des symboles pour les différents composants et les fils électriques qui relient ces composants (Figure 15.1).

Figure 15.1 : Deux symboles de circuit pour représenter un interrupteur.

Le symbole d'un contacteur est très simple (voir dans la figure précédente). Les deux principaux types de contacteurs sont l'interrupteur ou contacteur simple, et le commutateur ou contacteur double. Dans le premier, soit le circuit est fermé, soit il est ouvert, en fonction de la position du contacteur. Dans le commutateur, le contact est établi soit d'un côté, soit de l'autre. Ce composant permet de commuter la suite du circuit par rapport à l'arrivée du côté gauche.

On appelle également ce composant un contacteur double parce que le contacteur permet d'établir le circuit avec un terminal ou l'autre par rapport au contact commun. Les figures sont assez explicites. Sachez cependant que nous utilisons le même symbole, quel que soit l'aspect physique du composant. La Figure 15.2 donne quelques exemples de la variété de formats physiques des interrupteurs et commutateurs.

La Figure 15.3 réunit les symboles pour une batterie d'accumulateurs, une lampe et une résistance. Remarquez qu'il y a deux symboles différents pour la résistance : une est valable aux U.S.A. et l'autre en Europe. Au Royaume-Uni, les deux sont utilisées.

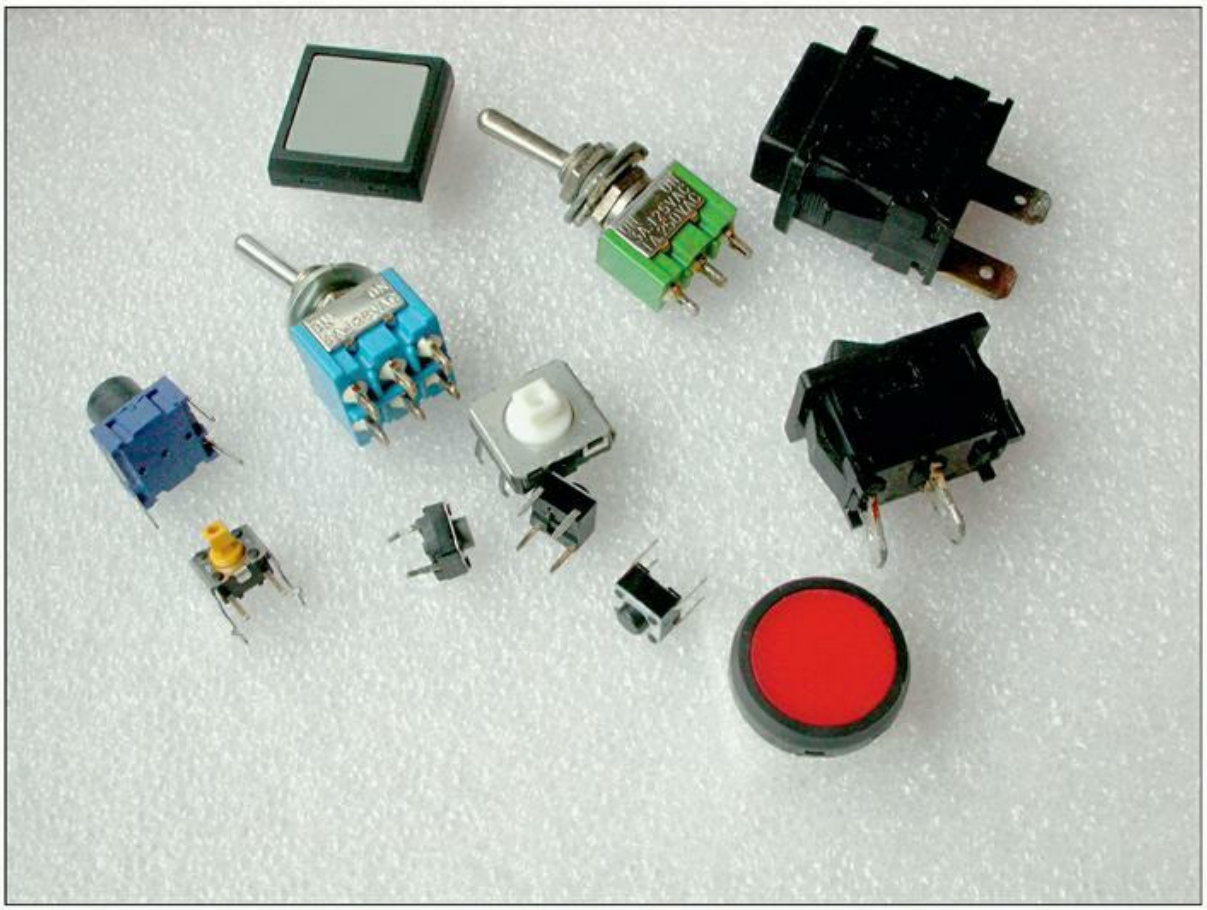


Figure 15.2 : Quelques exemples de commutateurs.

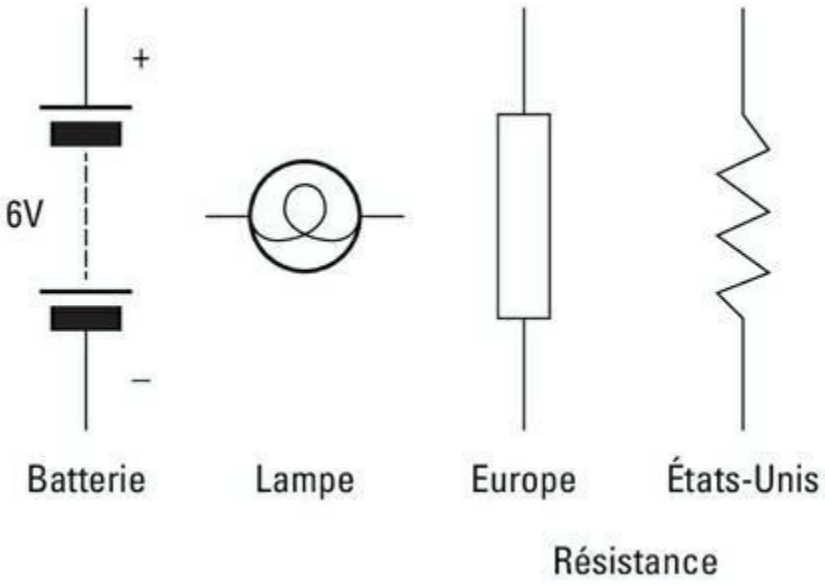


Figure 15.3 : Quelques symboles de schéma pour des composants.

La Figure 15.4 vous propose le circuit électronique le plus simple qui soit. Quand l'interrupteur est ouvert, le circuit n'étant pas fermé, aucun courant ne peut circuler, et la lampe ne peut pas s'allumer.

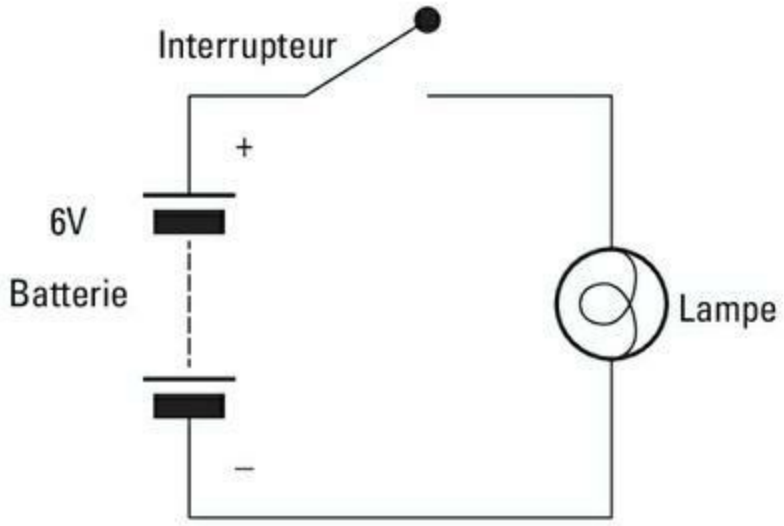


Figure 15.4 : Schéma d'un circuit très simple.

Quand nous refermons l'interrupteur comme dans la Figure 15.5, le courant peut s'écouler et en profiter pour allumer la lampe. Vous remarquez que le symbole de l'interrupteur est différent de celui de la Figure 15.4. Vous pouvez ainsi comprendre l'état dans lequel est le circuit. En temps normal, vous imaginerez l'interrupteur dans la position ouverte ou fermée et les conséquences de cette action sur le circuit. Il s'agit ici d'un circuit série, parce que tous les composants sont les uns à la suite des autres, et le même courant circule dans tous les éléments du circuit.

Dans un tel circuit, il n'y a qu'une seule valeur de courant. Lorsque le commutateur est fermé, le courant circule du positif de la batterie à travers le commutateur, puis la lampe qui s'allume pour revenir au pôle négatif de la batterie. Vous constatez que les électrons reviennent dans la batterie. Elle se vide parce qu'elle consomme de l'énergie pour déplacer les électrons entre les deux pôles. Les pôles + et – de la batterie montrent que le courant va s'écouler du + vers le –. Ce circuit ne comporte qu'une lampe qui est insensible au sens du courant, mais en électronique, ce sera rarement le cas. Généralement, il faudra veiller à envoyer l'énergie dans le bon sens.

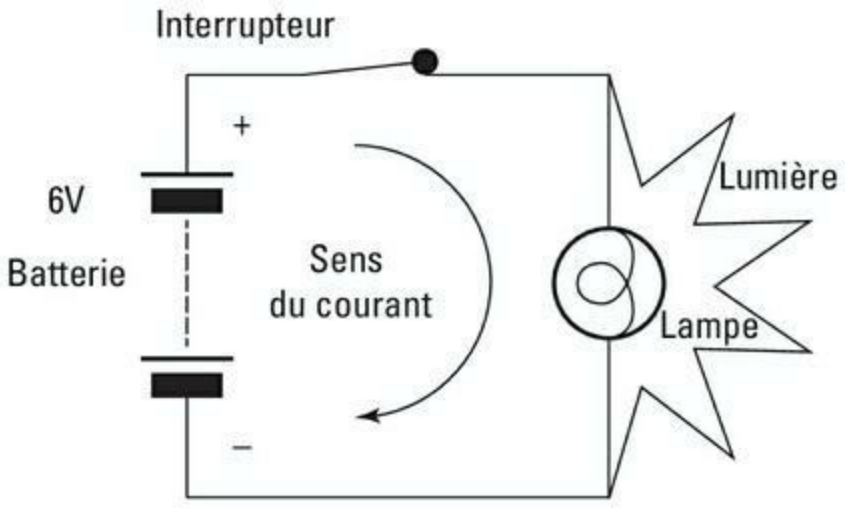


Figure 15.5 : Schéma du circuit simple après fermeture de l'interrupteur.

Représentation des circuits électroniques

Vous avez tout intérêt à apprendre à utiliser les symboles standard des composants dans vos schémas parce que cela constitue un langage universel et permet de bien comprendre comment fonctionne un circuit. Trop de gens perdent du temps à faire des dessins montrant l'aspect physique des composants et leurs connexions. Cela pourrait sembler intéressant au départ, mais ce genre de représentation physique devient inutilisable dès que les circuits sont un peu plus complexes. Prévoyez de passer un peu de temps à apprendre à lire les symboles. Les schémas normalisés constituent un moyen beaucoup plus efficace de concevoir un circuit.

S'il en fallait une preuve de plus, j'avais rapporté d'un voyage en Russie un kit de construction électronique pour mon fils. La documentation était uniquement en russe, que ni le père ni le fils ne comprenaient, mais le schéma utilisant la convention internationale, tout était parfaitement compréhensible.

Différents symboles sont possibles pour les unités de résistance. Vous rencontrerez 18 ohms, 18 Ω, ou comme nous allons l'écrire dans la suite de ce livre, 18 R.

Les unités des calculs sont le volt, l'ampère et l'ohm. En pratique, un ampère représente beaucoup de courant. En électronique, on parlera plus souvent en milliampères, abrégés en mA. Il y a bien sûr 1000 mA dans 1 A. De même, 1000 R correspondent à 1 kohm ou 1 kΩ.

Calcul des valeurs dans un circuit

Le schéma de la Figure 15.5 convient en termes pratiques, car il décrit ce qu'il faut câbler. En revanche, il ne suffit pas à effectuer des calculs avec la loi l'Ohm parce qu'il ne montre pas les résistances invisibles. En effet, tout composant physique possède une *résistance interne*. Le schéma qui la représente est un *circuit équivalent* (voir la Figure 15.6). Même les fils de connexion ont une résistance en série. Cette résistance peut être représentée comme placée en série avec le composant. Parfois, cette résistance interne est négligeable, mais pas toujours. Toute l'astuce est de savoir quand on peut l'ignorer.

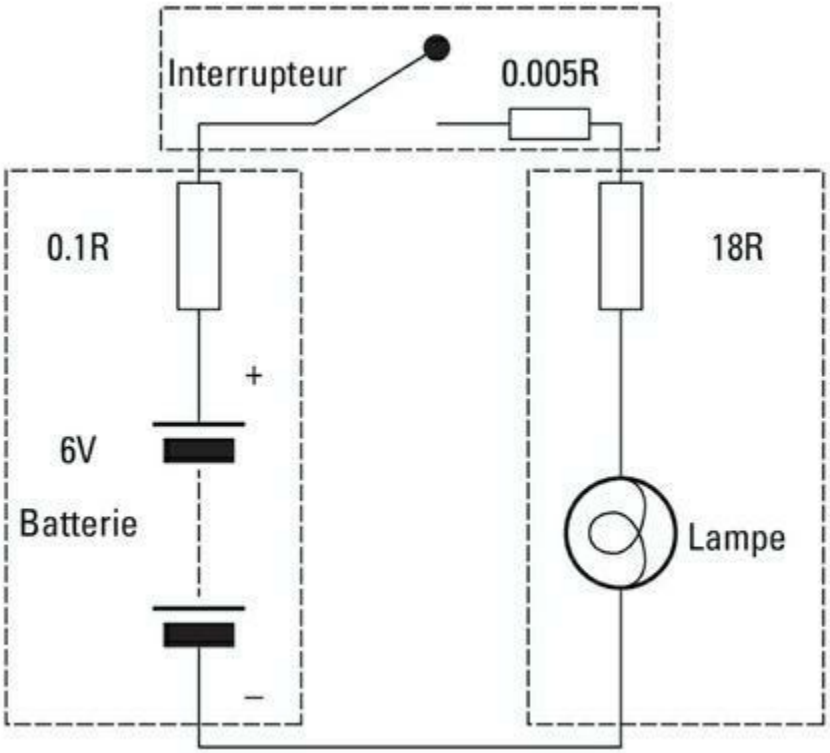


Figure 15.6 : Circuit équivalent avec les résistances internes des composants.

Lorsque plusieurs résistances sont placées en série, c'est-à-dire les unes à la suite des autres, la résistance totale est la somme de toutes les résistances. Dans la Figure 15.6, nous pouvons additionner toutes les valeurs des résistances, et nous obtenons 18R105 (c'est-à-dire 18,105 ohms). Vous constatez que la valeur de résistance la plus importante est celle de la lampe. La résistance de l'interrupteur est négligeable, et celle de la batterie également. Nous verrons dans le Chapitre 17 que ce n'est pas toujours le cas. Nous pouvons prendre en compte une résistance totale de 18R avec une tension de 6V pour trouver le courant qui va circuler dans le circuit :

```
I = E / R -->
Courant = 6 / 18 = 0,333 Amps soit 333 mA
```

Contraintes techniques des composants

Comment connaître la résistance d'un composant ? En général, elle est indiquée dans la *spécification* ou fiche technique du composant, que l'on appelle *Data sheet*. Rédigée par le fabricant, cette documentation donne des détails sur les domaines d'emploi du composant et toutes ses propriétés. Mais la valeur de la résistance n'est pas toujours clairement mentionnée. Dans le cas d'une lampe par exemple, les deux principaux paramètres sont la tension et le courant. Nous pouvons trouver dans le commerce une lampe 6V 0,33A. Cela peut nous suffire à trouver la résistance avec la loi d'Ohm. D'autres lampes, notamment celles de grande puissance, indiquent une puissance en watts. La puissance est le résultat de la multiplication du courant par la tension.

Un point remarquable est qu'une lampe n'a pas une résistance constante. C'est un composant *non linéaire*, car sa résistance change en fonction de la quantité de courant qui traverse la lampe. En effet, une lampe classique est constituée de quelques spirales de fil nu. Le courant qui passe dans le fil échauffe celui-ci, ce qui augmente la résistance et limite le courant. On arrive ainsi à un point d'équilibre lorsque la température a fait croître la résistance à tel point que le courant ne peut plus augmenter. Celui-ci se stabilise à la valeur prévue en fonction de la tension nominale. Nous utiliserons le concept de résistance non linéaire dans le Chapitre 17 pour calculer la résistance que nous devrons associer à une diode LED.

Pour les valeurs de tension et de résistance qui ne sont pas entières, le point décimal ou la virgule est parfois remplacé par l'abréviation de l'unité de mesure. 3,3V devient 3V3 et 4,7K devient 4K7. Cette convention permet d'éviter toute erreur lorsque le point ou la virgule est malaisé à lire en petits caractères.

La résistance interne de la source d'énergie qu'est la batterie ou n'importe quelle alimentation est un point essentiel parce qu'elle limite le courant que cette source peut fournir. Pour une batterie, cela dépend du processus chimique de décharge, mais cela ne nous empêche pas de considérer une résistance interne théorique. Une batterie capable de fournir beaucoup de courant a une faible résistance interne. On parle également parfois d'*impédance de sortie* de la batterie.

Pour l'instant, ces derniers concepts semblent n'avoir aucun rapport avec le Raspberry Pi. Vous allez voir dans les chapitres suivants que ce sont les concepts qu'il faut maîtriser pour que votre circuit puisse bien interagir avec le monde extérieur (en dehors du clavier et de l'écran).

Test d’un circuit avec un simulateur

Il existe de nos jours des simulateurs qui permettent de tester un circuit avant même de le construire. C’est une excellente idée pour vérifier que vous n’avez pas fait d’erreur. Cela dit, certains de ces simulateurs réclament une phase d’apprentissage alors que les autres utilisent des composants théoriques et non des composants réels, ce qui peut fausser quelque peu les résultats. Globalement, ce sont tous de très bons outils. Il existe un simulateur gratuit conçu pour le Raspberry Pi. Pour en savoir plus, visitez la page *www.raspberrypi.org/archives/1917*.

Le connecteur GPIO

Le Raspberry Pi se fonde sur la puce BCM2835. Il s’agit d’un SOC (*System On Chip*) qui est conçu pour être exploité dans les systèmes embarqués, à la différence des microprocesseurs de nos ordinateurs. Un *système embarqué* contient un ordinateur, mais il n’est pas utilisé en tant qu’ordinateur. C’est le cas des smartphones, des lecteurs multimédias portables et des boîtiers Internet. Ce genre de circuit offre un certain nombre de connexions qui permettent avec le logiciel approprié de contrôler des composants réels comme des boutons, des afficheurs, un haut-parleur, *etc*. En théorie, le circuit BCM2835 offre 54 signaux qui portent le nom collectif de GPIO (*General Purpose Input/Output*) et peuvent être contrôlés par logiciel. Certains des signaux sont déjà utilisés pour contrôler les périphériques qui font du BCM 2835 un vrai ordinateur : le lecteur de carte SD, le port USB et le port Ethernet. Les autres signaux sont disponibles.

Au lieu de les ignorer, les concepteurs du Raspberry Pi ont choisi de rendre ces signaux GPIO supplémentaires accessibles sur un connecteur qu’ils ont appelé **P1**. C’est un vrai bonus. Ce connecteur distingue le Raspberry des ordinateurs habituels. Cela dit, tous les signaux et broches correspondantes n’ont pas été rendus accessibles sur le connecteur. Certains sont destinés par exemple au connecteur de la webcam, et d’autres ne sont même pas ramenés à l’extérieur de la puce. Il faut savoir que la puce BCM2835 est un boîtier soudé sous forme d’une matrice de broches au format BGA (*Ball Grid Array*), chaque petite bille de soudure étant séparée des autres de moins d’un millimètre. L’espace est tellement ténu que vous ne pouvez faire passer qu’une seule piste de circuit imprimé entre deux connexions de la puce.

Pour pouvoir amener vers l’extérieur toutes les broches d’une puce telle que ce BCM, il est obligatoire de concevoir un circuit imprimé multicouche. Lorsque vous regardez votre circuit imprimé Pi, vous

pourriez croire qu'il n'y a que deux faces, dessus et dessous. En réalité, il y a plusieurs couches de piste prises en sandwich, en l'occurrence elles sont au nombre de 6.

Même avec toutes ces couches, il n'y a pas assez de place pour faire sortir les 54 signaux du GPIO depuis le processeur. Rajouter plus de couches aurait augmenté encore le prix de la fabrication du circuit ; les signaux supplémentaires ne sont donc pas disponibles sans ce surcoût. Vous savez bien sûr que le prix du Pi est un de ses grands avantages. Un juste milieu a été trouvé, puisque 17 des broches supplémentaires ont été rendues accessibles sur le connecteur P1. D'ailleurs, ce nombre pourra augmenter dans les révisions futures de la carte. Les deux premières révisions ont déjà une combinaison de broches CPIO différente. Les cartes plus récentes de la révision 2 (Model B) comportent un second connecteur plus petit, donnant accès à 4 broches supplémentaires du GPIO. Nous verrons exactement quelles sont les broches et quels sont ces signaux dans le chapitre suivant.

Usage général ou spécifique

GPIO est l'abréviation anglaise de *General Purpose Input/Output*, qui signifie *broches d'entrée/sortie à usage général*. L'expression « usage général » souligne que les broches ne sont pas dédiées à certaines fonctions, mais qu'elles dépendent simplement du programme qui va les contrôler. Ce sont des broches d'entrée/ sortie, parce que chacune d'elles peut être configurée par le logiciel pour servir soit en entrée, soit en sortie. Une broche d'entrée permet au programme de lire la tension qui s'y trouve (soit tension haute, soit tension basse). Inversement, lorsqu'une broche est configurée en sortie, le programme peut décider si elle va délivrer une tension haute ou basse. De nombreuses broches ont des pouvoirs particuliers ou des modes d'utilisation spécifiques. Cela correspond à des fonctions spécialisées qui peuvent être exécutées directement sans logiciel. Ce genre d'usage revient à entrer directement dans les profondeurs du processeur. Lorsqu'une broche bascule dans l'un de ces modes, elle n'est plus une broche à usage général. Une broche peut par exemple servir à émettre en sortie un flux continu de passages entre tension haute et tension basse, sans réclamer d'intervention d'un programme. Cela permet par exemple de brancher une broche de sortie à un haut-parleur pour entendre un son généré à une certaine fréquence, et cela jusqu'à ce que vous demandiez l'arrêt de la fonction.

Mais nous allons trop loin. Pour l'instant, intéressons-nous à l'usage général de toutes ces broches.

Étude d'une broche GPIO

Les broches et signaux GPIO permettent d'entrer en interaction avec le monde extérieur. Leur principe est simple.

La Figure 15.7 donne le schéma du circuit équivalent d'une broche GPIO du Raspberry Pi configurée en sortie. Vous constatez que cette broche joue le même rôle qu'un commutateur pouvant établir le contact soit avec la source de tension de 3V3, soit avec la masse (le 0V). La masse est également appelée le *point commun*, le point froid ou encore la *référence*. Elle sert de repère et de point commun pour toutes les mesures dans un circuit. La masse correspond au pôle négatif de la batterie ou de l'alimentation. Vous constatez qu'entre le commutateur et la broche de sortie se trouve une résistance en série. Cette résistance va limiter le courant qui pourra circuler dans la broche de sortie.

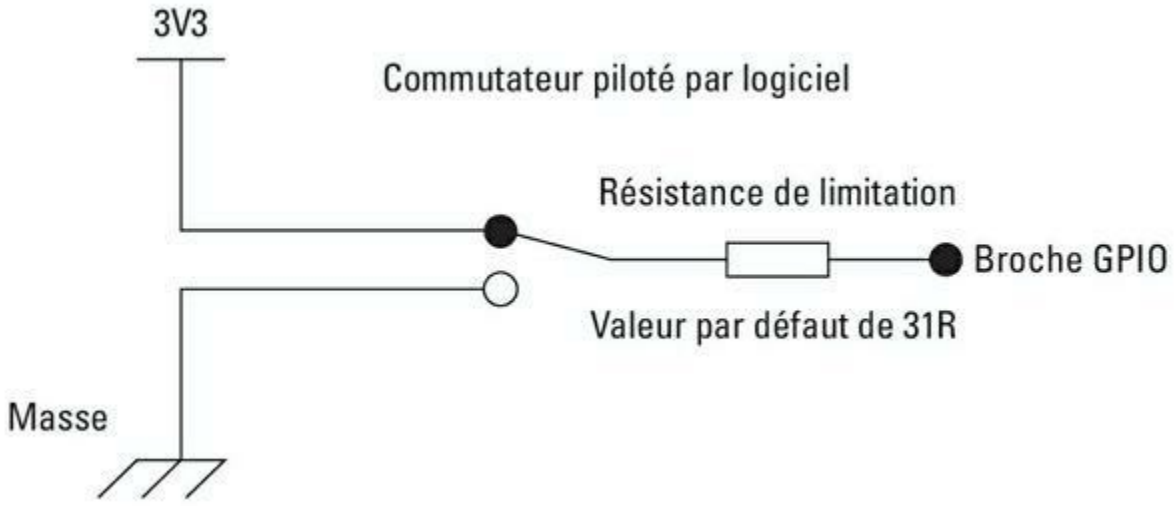


Figure 15.7 : Une broche GPIO utilisée en sortie.

TROUVER UNE VALEUR DE COURANT SÛRE

Il existe un plafond de courant au-delà duquel vous êtes assuré de détruire au minimum les circuits de la broche de sortie, voire la totalité de votre Raspberry. Une valeur un peu inférieure ne va rien détruire, mais va faire vieillir prématurément le circuit et le faire tomber en panne plus tôt que prévu. Pour un courant inférieur, vous réduisez la fatigue de la matière, et en descendant encore, vous arrivez à un courant sûr. Hélas, la valeur de courant sûre n'est pas connue pour le circuit utilisé dans le Raspberry. D'ailleurs, cette valeur

n’est pas connue en général. Il vaut donc mieux se limiter à une valeur de précaution, voire un peu inférieure. Méfiez-vous des gens qui prétendent que leur circuit est capable de supporter 30 mA sur une broche sans problème. Ces gens confondent une broche détruite avec une broche qui va bientôt l’être. C’est un peu comme fumer : vous ne mourrez pas immédiatement, mais cela vous endommage, et pourra vous emporter si une autre cause ne se présente pas avant. Personne ne peut prétendre que fumer est sans danger.

Dans le Raspberry, la valeur de cette résistance de broche peut être réglée sur une plage assez réduite. Par défaut, cette valeur est égale à 31R, mais cette valeur ne suffit pas à protéger le circuit si vous ne prévoyez pas une résistance extérieure dans le circuit branché à la broche. Une broche de sortie peut *a priori* commuter entre deux tensions : soit 0V, soit 3V3. Ces deux tensions correspondent aux deux niveaux logiques qui portent différents noms : haut et bas, vrai et faux ou 0 et 1.

Ce fonctionnement en tout ou rien (binaire), basé sur deux tensions différentes est simple à comprendre, mais le courant que peut fournir une broche de sortie est limité à environ 16 mA. Cela correspond au courant maximal que le Pi devrait fournir à la charge branchée en sortie, mais ce n’est pas le courant qu’il fournira, car cette valeur va dépendre de la résistance de la charge connectée. J’ai cité une valeur de 16 mA, mais nous entrons ici dans une zone un peu floue. Cette valeur est considérée comme sans danger pour le Pi, mais cela n’implique pas qu’une valeur de 17 mA va le détruire.

Utilisation pratique d’une broche de sortie

Que peut-on faire d’un tel commutateur en sortie ? Soit nous alimentons une charge qui consomme très peu de courant, soit nous pilotons un composant qui sera capable de supporter un courant plus important. Si cela ne vous dit rien, sachez que c’est une des utilisations principales de l’informatique physique. En effet, la charge peut être une lampe, un moteur électrique, une bobine ou un solénoïde (qui permet de faire basculer un relais) ou toute autre chose qui utilise de l’électricité, donc quasiment tous les objets du monde moderne. Tout objet qui utilise de l’électricité peut être contrôlé.

Voyons comment contrôler une lumière, mais pas une de ces ampoules que nous avons au plafond dans nos maisons. Découvrons la diode électroluminescente pour laquelle on utilise habituellement l'acronyme anglais LED (*Light-Emeting Diode*). Une LED s'illumine avec très peu de courant : les 16 mA que nous pouvons offrir suffiront largement. Nous devons même limiter le courant à moins de 10 mA en ajoutant une résistance en série de 330R. Comment avons-nous trouvé cette valeur ? Nous expliquerons en détail ces calculs dans le Chapitre 17.

Pour l'instant, concentrons-nous sur les deux circuits de la Figure 15.8. Vous pouvez y voir deux manières de connecter une diode LED, ou n'importe quelle autre charge, à une broche du connecteur GPIO. Nous n'avons montré ici que la broche, sans montrer la résistance équivalente de la source de puissance dont nous avons parlé plus haut. En effet, avec une résistance externe de 330R, la résistance interne de 31R devient négligeable.

Il y a deux manières de connecter une charge. La première (à gauche) est une *source de courant*, c'est la méthode qu'un débutant considère comme normale. Lorsque la broche de sortie GPIO est forcée par un programme pour délivrer une tension haute (c'est-à-dire qu'elle délivre du 3V3), le courant vient de la broche (c'est la source), traverse la LED et la résistance pour rejoindre la masse et reboucler le circuit. Puisque le circuit est fermé, la LED brille. Lorsque le programme force la broche à délivrer un état bas (le commutateur est relié au 0V ou à la masse), aucun courant ne peut circuler et la LED ne brille pas. Cette technique s'appelle la source de courant parce que le courant provient de la broche GPIO qui sert de borne positive.

L'autre technique (partie droite de la Figure 15.8) correspond à un *puits de courant* (*current sink*). Lorsque le programme force la broche de sortie GPIO à 0V, le courant peut circuler à travers la LED et la résistance pour rejoindre la masse par la broche. Pour éteindre la LED, il suffit de délivrer la tension haute sur la broche de sortie. Dans ce cas, aucun courant ne peut circuler puisque les deux extrémités (la LED et la résistance) sont toutes deux au potentiel de 3V3. Comme il n'y a pas de différence de potentiel, le courant ne peut pas circuler.

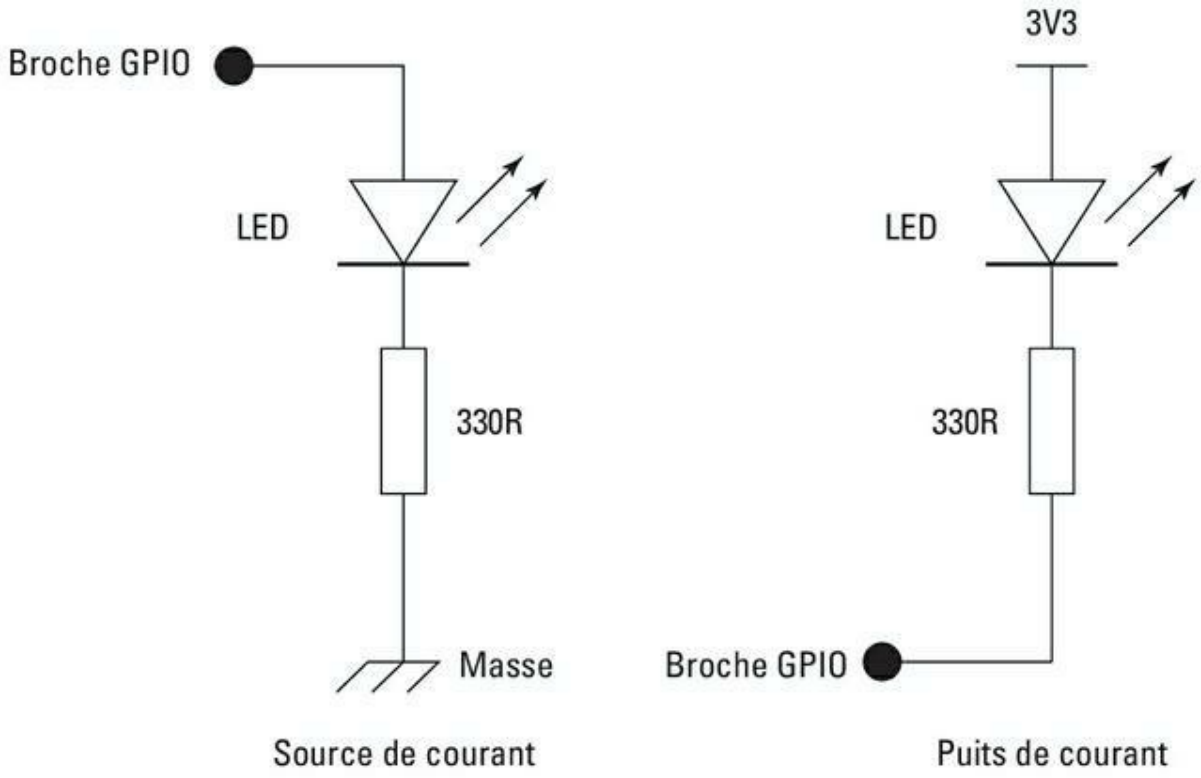


Figure 15.8 : Les deux techniques de pilotage d'une charge LED.

Notez que les positions de la résistance et de la LED peuvent être interverties sans aucun effet. Ces deux techniques peuvent être imaginées comme un interrupteur de Plus et un interrupteur de masse. Nous y reviendrons quand nous construirons des projets dans les Chapitres 17 et 18.

Utilisation d'une broche GPIO en entrée

L'autre mode élémentaire d'utilisation d'une broche du GPIO est en entrée. Dans ce mode, vous n'avez plus à vous soucier de la quantité de courant parce qu'une broche programmée en entrée possède une très forte *impédance d'entrée* (sa résistance interne est très élevée). Cette résistance élevée empêche la circulation de l'électricité. Pour être précis, l'impédance est un peu plus complexe que la résistance, mais pour l'instant, vous pouvez confondre les deux termes, et ils sont tous deux mesurés en ohms.

La résistance est une propriété statique d'un matériel alors que l'impédance est une propriété d'un circuit qui tient compte de son comportement variable avec du courant alternatif. Du fait qu'une entrée possède une très forte impédance, il est quasiment impossible que du courant puisse circuler à travers elle. Voilà pourquoi on peut connecter

une telle entrée directement au 0V ou au 3V3 sans prévoir de résistance de limitation. D'ailleurs, l'impédance est tellement élevée que si vous ne connectez rien, vous captez de très faibles ondes radio et parasites, ce qui risque de faire apparaître des valeurs aléatoires lorsque vous lisez l'état de la broche.

En effet, même la simple présence du corps humain au contact ou juste à côté d'une entrée à haute impédance peut jouer le rôle d'une antenne et créer des erreurs de mesure. Les débutants en sont souvent étonnés, et croient qu'ils ont découvert un phénomène mystérieux. Pas du tout. Les très faibles quantités d'énergie des ondes radio de l'environnement ne sont pas absorbées de la même manière dans un circuit à haute impédance et dans un circuit à basse impédance. Dans ce dernier, le parasite entraînerait une circulation de courant, mais la très faible puissance résulterait en une tension vraiment minuscule. Notez qu'il suffit d'un câble transportant du courant alternatif (un câble secteur par exemple) dans les parages pour que ce câble émette des interférences radioélectriques.

Voyons pourquoi une entrée à haute impédance est sensible aux parasites. Supposons qu'une interférence ayant une tension de 2V suffise à perturber le signal d'un circuit et à entraîner un mauvais fonctionnement. Si l'impédance est faible, par exemple 1 kR, il faudrait pour établir une tension de 2V un courant de 2 mA (loi d'Ohm). Cela correspond à une puissance (tension multipliée par courant) de 4 mW. Mais si la résistance d'entrée est de 1 million d'ohms, vous pouvez obtenir un pic de tension de 2V avec un courant circulant de seulement 2 µA (2 millionième d'ampère). Cela correspond à une puissance de 4 µW. Voilà pourquoi une résistance importance est beaucoup plus sensible aux interférences : elle requiert moins de puissance de la part de la source parasite pour faire apparaître une tension significative. Des champs d'interférence faibles peuvent donc générer assez de tension pour perturber le fonctionnement d'un circuit.

Nous pouvons en conclure qu'il faut prendre une précaution importante avec les entrées et ne jamais les laisser libres (flottantes). Elles doivent être forcées à une tension ou une autre, c'est-à-dire soit à 3V3, soit à 0V. Si vous connectez une broche d'entrée à la broche de sortie d'un autre circuit, cela n'est pas nécessaire. En revanche, si vous voulez détecter si un interrupteur est ouvert ou fermé, il faut forcer l'état de la broche d'entrée. Normalement, il suffit de relier cette broche au moyen d'une résistance soit à la tension haute 3V3, soit à la masse (0V).

Une résistance qui sert à ce forçage se nomme une résistance de rappel haut ou bas (de tirage, *pull-up* ou *pull-down*) comme le montre la Figure 15.9. La résistance de rappel haut (vers le 3V3) est préférable,

parce que les commutateurs sont souvent à l'extrémité de câbles assez longs, et il est préférable que le câble soit à la masse plutôt qu'au 3V3. (Les risques de dégâts sont moindres si vous connectez par erreur un fil qui est à la masse plutôt qu'un fil qui est à la tension haute.) Cette technique consistant à utiliser des résistances de rappel est tellement fréquente que le processeur dans votre Pi dispose de résistances intégrées. Par programme, vous pouvez connecter ou activer ces résistances internes de rappel vers le haut ou vers le bas. Nous verrons dans le Chapitre 16 comment contrôler ces résistances par logiciel.

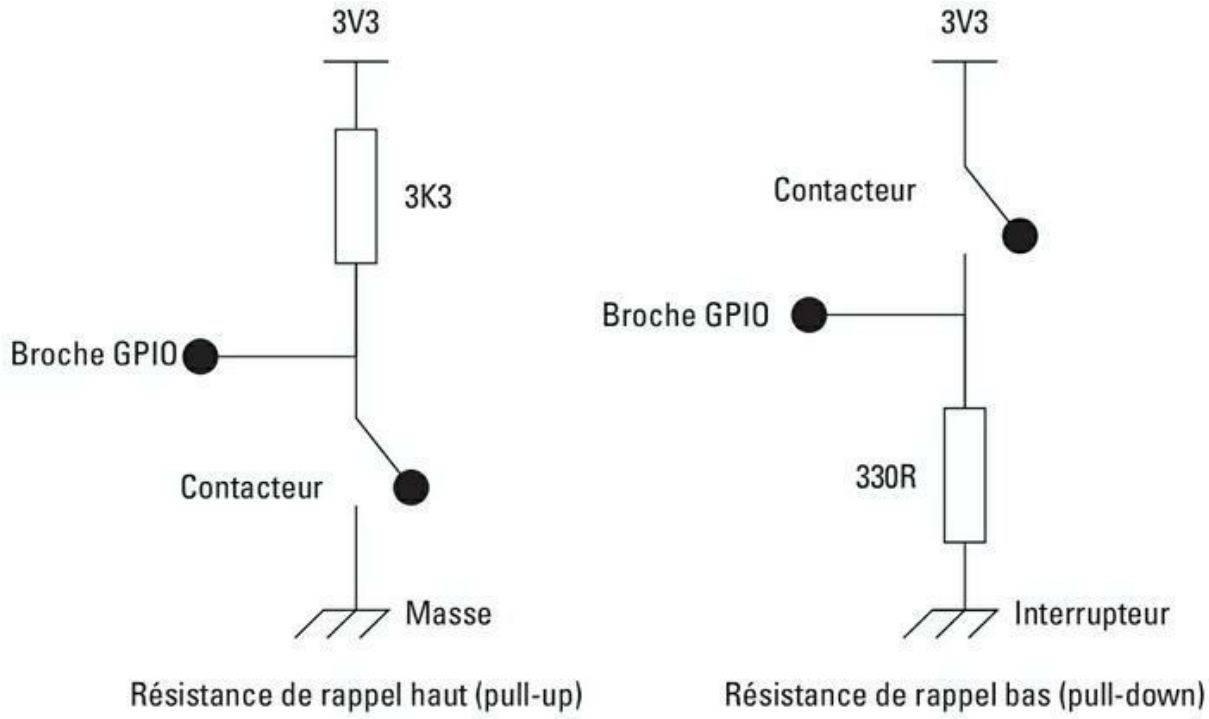


Figure 15.9 : Résistances de rappel haut et bas pour stabiliser une broche d'entrée GPIO.

Découverte du fer à souder

Bien qu'il soit possible de construire un circuit sans soudure, pour travailler sérieusement, il faudra en passer par là. Souvent, le débutant commence à paniquer, mais il suffit de savoir que même un enfant peut réussir à bien souder. D'ailleurs, le fils de l'auteur a reçu son premier fer à souder à 9 ans et a appris tout seul. La soudure suppose deux éléments : la *soudure* en elle-même qui est un alliage de deux métaux au minimum et le *flux* qui est un décapant chimique. Lorsqu'il s'agit de souder un tuyau de gaz, le flux doit être appliqué autour du joint, avant de chauffer le joint avec un chalumeau pour appliquer la baguette de soudure. Le rôle du flux chauffé est de nettoyer la surface et de faire couler la soudure. Il y

parvient en réduisant la tension de surface de la soudure fondue. Sans flux, la soudure ferait des boules à cause de la tension de surface.

Même l'eau du robinet possède une tension de surface. Pour la réduire, nous nous servons de savon, et c'est ce qui permet à l'eau de mouiller les choses. Le savon permet à l'eau de mouiller les choses. Il est impossible d'utiliser du savon avec de la soudure parce que la chaleur détruirait les molécules de savon. La plupart des flux pour les travaux lourds utilisent des produits chimiques très corrosifs comme l'acide hydrochlorique ou phosphorique. Ce sont des produits trop corrosifs pour l'électronique. On utilise plutôt une sorte de résine vendue en pots, mais il est bien plus pratique d'utiliser de la soudure multi-âme, car le flux est directement intégré au fil de soudure sous forme de cinq très fines bandes. Vous êtes ainsi assuré que la quantité de flux apportée est toujours appropriée à la quantité de soudure que vous faites fondre.

Nous vous conseillons d'utiliser un alliage de soudure étain/plomb à 60/40, d'un diamètre de 0,7 millimètre et doté d'un flux de résine. Tout autre choix vous rendrait la vie difficile. Nous avons par exemple remarqué que les flux autonettoyants ou « sans fumée » sont moins faciles à utiliser et plus coûteux. Associez votre bon fil de soudure à un bon fer à souder, si possible avec contrôle de température et une panne fine.

LA SOUDURE SANS PLOMB

La directive européenne RoHS (*Reduction of Hazardous Substances*) vise à interdire l'utilisation de certains métaux toxiques et plastiques dans les équipements électriques et électroniques. Le principal métal visé est le plomb, mais ce n'est pas le seul. Vous pouvez acheter de la soudure sans plomb, mais elle coûte plus cher parce que le plomb est remplacé par de l'argent. De plus, cette soudure est moins facile à utiliser et le résultat a une durée de vie inférieure. Le fer à souder doit être plus chaud, ce qui peut plus facilement endommager les composants à souder.

Cette soudure est également moins mouillante : elle épouse moins bien la forme du joint et provoque plus facilement la création de barbillons de soudure, ce qui peut causer des courts-circuits quelques années plus tard. Mais heureusement, l'électronique de loisir n'est pas obligée d'utiliser de la soudure sans plomb, ni aux U.S.A., ni en

Europe. D'ailleurs, il n'y a pas d'effets mesurables sur la santé lorsque l'on utilise de la soudure au plomb. La directive RoHS a pour principal objectif de limiter l'accumulation de plomb sur les sites de recyclage, car cela pollue les nappes phréatiques. En revanche, si vous comptez commercialiser les circuits électroniques que vous fabriquez avec de la soudure au plomb, sachez que c'est interdit. C'est un peu comme les microbrasseries domestiques : vous pouvez fabriquer autant de bière que vous voulez, mais vous ne pouvez pas la vendre. Notez qu'il est malgré tout conseillé de se laver les mains après toute opération de soudure et d'éviter bien sûr de mettre le fil de soudure dans la bouche.

Certains conseillent d'utiliser du vieux matériel ou du matériel bas de gamme pour commencer, afin d'acquérir un bon outil seulement une fois que l'on est dégrossi. Ce n'est pas une bonne idée ! En tant que débutant, vous devez vous battre pour réussir, mais il ne faut pas ajouter à la difficulté en utilisant de mauvais outils. Un bon fer comporte une base pour le reposer et un évidement pour y mettre une éponge. Choisissez une bonne éponge naturelle pour qu'elle ne fonde pas au contact du fer. N'utilisez pas une éponge synthétique que votre fer va traverser quand vous voudrez le nettoyer.

Vos premières soudures

Pour bien réussir vos soudures, commencez par les préparer en établissant une bonne liaison mécanique. Lorsque vous devez par exemple souder deux fils ensemble, repliez l'extrémité de chacun des deux en forme de crochet et torsadez-les à la main ou avec une pince pliante pour former un oeillet ou une épissure.

Prenez votre fer et essuyez la panne sur l'éponge humide puis faites fondre un peu de soudure sur la pointe. Vous facilitez ainsi le transfert de chaleur entre la panne et le point de soudure. Approchez alors simultanément le fer, le fil à souder et les deux fils à relier. Observez bien le joint et la soudure pour voir comment elle coule. Retirez le fil de soudure, mais laissez le fer en place jusqu'à voir la soudure couler autour du joint et pénétrer dans les évidements. C'est à ce moment-là que le joint est assez chaud pour que vous puissiez enlever le fer. Ce geste est assez rapide et réclame un peu d'entraînement.

Les débutants font souvent l'erreur de trop charger la soudure. Utilisez toujours le moins de fil à souder possible. Il arrive rarement que la soudure soit défectueuse parce qu'il n'y a pas eu assez de matière, mais bien plus souvent parce qu'il y en a eu trop. Lorsque vous avez terminé, vous devez voir quelques restes de flux noir autour de la pointe du fer. Essuyez le fer sur l'éponge avant de le reposer sur son support. Ne faites pas bouger le point de soudure en attendant qu'il refroidisse. Si votre fer est de bonne qualité, il est prêt pour la soudure suivante. Un mauvais fer réclamera quelques secondes pour remonter en température.

Il n'est pas inutile de prévoir un système pour extraire les fumées du point de travail. Un ventilateur peut suffire à éloigner les fumées de votre visage. En effet, la chaleur de votre corps a tendance à attirer les fumées vers vous. Ne les respirez pas, surtout si vous passez beaucoup de temps (plusieurs heures) à faire des soudures. Notez que ces fumées proviennent du flux décapant ; elles ne contiennent pas de plomb.

Nous avons déjà dit que le projet du Chapitre 16 ne nécessitait aucune soudure. Mais ce genre de projet est rare. Dans les deux autres chapitres de cette partie, tous les projets vont vous demander d'avoir appris à maîtriser un fer à souder.

N. d. T. : Pour être exact, ce n'est pas de la soudure, mais du brasage. En effet, souder suppose de faire fondre les deux parties à réunir, ce qu'il ne faut certainement pas faire vu la fragilité en température des composants électroniques. Le brasage suppose de faire fondre un métal qui va réunir deux surfaces à relier.

Cartes d'extensions du commerce

Il existe un certain nombre de cartes d'extension et d'interface pour le Raspberry Pi qui vous facilitent la vie puisqu'une partie du travail est déjà réalisée. Certaines cartes sont passives, et ne font que rendre plus confortable l'accès aux différentes broches du GPIO. D'autres cartes contiennent des composants et augmentent le nombre d'entrées et de sorties du Pi, ou proposent des fonctions spéciales qui ne sont pas directement disponibles, comme le contrôle de la luminosité pour les modèles de diodes LED courants. Vous pourrez toujours ajouter ces circuits complémentaires quand vous en ressentirez le besoin. Il est néanmoins toujours possible de créer soi-même toutes les fonctions correspondantes.

Sachez que ces cartes complémentaires ne sont pas indispensables, et peuvent sensiblement augmenter le coût de vos projets, surtout parce qu'elles offrent plus de possibilités que vous en avez besoin pour chaque

projet. Si vous comptez démonter votre projet une fois que vous l'avez utilisé, vous pourrez réutiliser ce genre de carte, mais si vous voulez garder vos projets tels quels pour pouvoir les montrer à vos amis, vous immobilisez une carte onéreuse. Mieux vaut dans ce cas construire le circuit correspondant vous-même. Tous les projets des deux chapitres suivants du livre sont autonomes et ne réclament aucune carte d'extension. Cela dit, certaines personnes pourront être intéressées par le gain de temps qu'elles procurent. De nouvelles cartes d'extension apparaissent sans cesse. Nous allons en découvrir deux dans cette section.

La carte Gertboard

La carte Gertboard est l'ancêtre de toutes les cartes d'extension pour notre circuit. C'est celle qui se rapproche le plus de l'idée d'une carte d'interface officielle pour le Raspberry Pi car elle a été conçue par Gert van Loo qui est un des membres de l'équipe ayant conçu le Pi. Pour autant, ce n'est pas un produit de la fondation Pi. C'est une carte très fournie, qui réunit plusieurs interfaces, et embarque même un processeur de style Arduino. Arduino est un contrôleur autonome très utilisé dans le monde de l'art et de l'ingénierie. En apparence, il ressemble au Raspberry Pi, mais c'est en réalité quelque chose de très différent. Le Arduino est bien meilleur que le Pi pour tout ce qui réclame un temps de réponse très bref et un travail en temps réel, mais il ne possède aucun affichage et ne peut être programmé qu'en C++ sur une autre machine. En revanche, combiner le Pi avec un Arduino permet des projets très intéressants, et c'est pourquoi Gert a décidé d'implanter un processeur Arduino sur sa carte d'extension. Cette carte est destinée à servir à l'enseignement, afin de tester différentes approches d'interfaces. C'est une carte entièrement terminée qu'il suffit de connecter à votre Pi. Citons ses principales caractéristiques (ne vous inquiétez pas si vous ne saisissez pas le sens de tous les termes) :

- douze ports d'entrées/sorties tamponnés par un circuit 74HC244 et chacun muni d'une LED ;
- trois boutons-poussoirs ;
- un MCP4802, convertisseur numérique/analogique à deux canaux sur 8 bits ;

- un ULN2803A, six canaux à collecteur ouvert acceptant jusqu'à 50V ~80mA/ canal ;
- un ATmega328P, microcontrôleur Atmel®AVR® 8 bits (Arduino) ;
- un L6203, circuit contrôleur de moteur électrique 48V 4A ;
- un 780xx 3V3, régulateur de tension abaisseur.

Tous ces éléments sont montés sur un circuit imprimé. La carte est livrée avec les cavaliers et câbles de liaison, un câble en nappe et des fiches pour les connexions vers le Pi. Décrire en détail toutes les fonctions offertes et leur utilisation occuperait un livre entier.

Il y a peu de chance qu'un projet ait besoin simultanément de toutes ces fonctions, et la plupart sont un peu trop complexes pour le débutant moyen. Mais lorsque vous irez plus loin que ce que nous présentons dans ce livre, vous serez peut-être intéressé par l'acquisition d'une telle carte.

Un point intéressant de la carte Gertboard est que toute la documentation est téléchargeable, ce qui vous permet d'étudier ce que vous pouvez en faire avant de décider de l'acquérir. Pour en savoir plus et télécharger cette documentation, rendez-vous à la page *www.raspberrypi.org/archives/1734*.

La carte Pi Face

La carte d'extension Pi Face a été conçue par une équipe de l'école de science informatique de l'université de Manchester ; elle est destinée au marché de l'éducation. Son prix est inférieur à celui de la Gertboard (environ 30 € au lieu de 50). Elle est beaucoup moins complète, mais réunit un certain nombre d'éléments qui vont seront utiles dans la plupart des projets simples. Elle comporte des diodes LED et boutons-poussoirs ainsi que deux relais (des commutateurs pilotés par électro-aimant) ce qui permet de contrôler des courants plus forts. Elle offre 8 entrées protégées et 8 sorties avec tampon, et l'ensemble est accessible par des borniers. Vous pouvez télécharger tous les documents et exemples depuis Google Documents (*https://docs.google.com/folder/d/0B-UAZ9CyJCLGQjJ3RDlqa2pqaDg/edit?pli=1*). L'évolution de la carte est décrite à l'adresse *www.raspberrypi.org/archives/tag/pi-face*. Il existe un livre en anglais qui présente des projets réalisés avec la Pi Face :

Raspberry Pi Projects par Dr. Andrew Robinson (publié chez John Wiley & Sons, Inc.).

Autres cartes d'extension

De nombreuses autres cartes sont produites par de petits fabricants, et certaines peuvent être construites en récupérant les documents sur le Web. Un bon point de départ pour en savoir plus à leur sujet est la page http://elinux.org/RPi_Expansion_Boards.

Quelques cartes d'extension

Il existe deux grandes catégories de cartes d'extension pour Raspberry Pi : les cartes statiques qui servent à faciliter l'accès aux broches du connecteur GPIO, et les cartes actives qui embarquent un certain nombre de composants soudés. Découvrons quelques-unes des cartes de la seconde catégorie, en laissant la présentation d'une carte de connexion au prochain chapitre.

Depuis que le Raspberry Pi a été lancé en 2012, de nombreuses sociétés ont conçu et commercialisé toutes sortes de cartes d'extension. En général, elles sont accompagnées de programmes d'exemple, et la plupart des utilisateurs s'en contentent. C'est bien dommage, car vous pouvez aller beaucoup plus loin dans l'utilisation de ce genre de carte qu'en vous limitant aux exemples fournis. Ces cartes actives réunissent en général plusieurs capteurs qui permettent à votre circuit de réagir aux conditions extérieures. De nouvelles cartes sont sans cesse mises sur le marché.

Pour être plus précis, il existe en fait trois types de cartes :

- Les cartes de connexion déjà citées, pour simplifier les connexions GPIO. Elles utilisent un câble en nappe ou des fils volants.
- Les cartes qui se branchent directement sur le connecteur GPIO et viennent recouvrir presque entièrement la carte RasPi. On les appelle souvent boucliers ou *shields*.
- Les cartes d'extension sophistiquées, qui ressemblent à des boucliers, mais qui ont fait

> l'objet d'une identification officielle. Souvent, ces cartes contiennent un logiciel qui permet à la carte RasPi de les identifier dès le démarrage et de configurer les broches GPIO automatiquement. Ces cartes portent le nom HAT, qui signifie *Hardware Attached on Top*. Pour en savoir plus à leur sujet, visitez cette page :
>
> http://www.raspberrypi.org/introducing-raspberry-pi-hats

La carte Sense HAT

Cette carte a été conçue spécialement pour la mission spatiale Astro Pi. Deux exemplaires construits avec un très grand soin ont été embarqués sur la station spatiale internationale ISS en décembre 2015 et ont exécuté du code écrit par des collégiens. La carte comporte une matrice de LED tricolores de 8 sur 8, un joystick à cinq boutons et une série de capteurs pour mesurer l'accélération, le magnétisme, la température, la pression et l'humidité, sans oublier un gyroscope (Figure 15.10). La carte est accompagnée d'une librairie de fonctions Python très complète qui permet de l'exploiter aisément. Vous trouverez une description détaillée de l'utilisation de cette carte sur le site Web de la fondation Raspberry Pi, à l'adresse suivante :

https://www.raspberrypi.org/learning/getting-started-with-the-sense-hat/

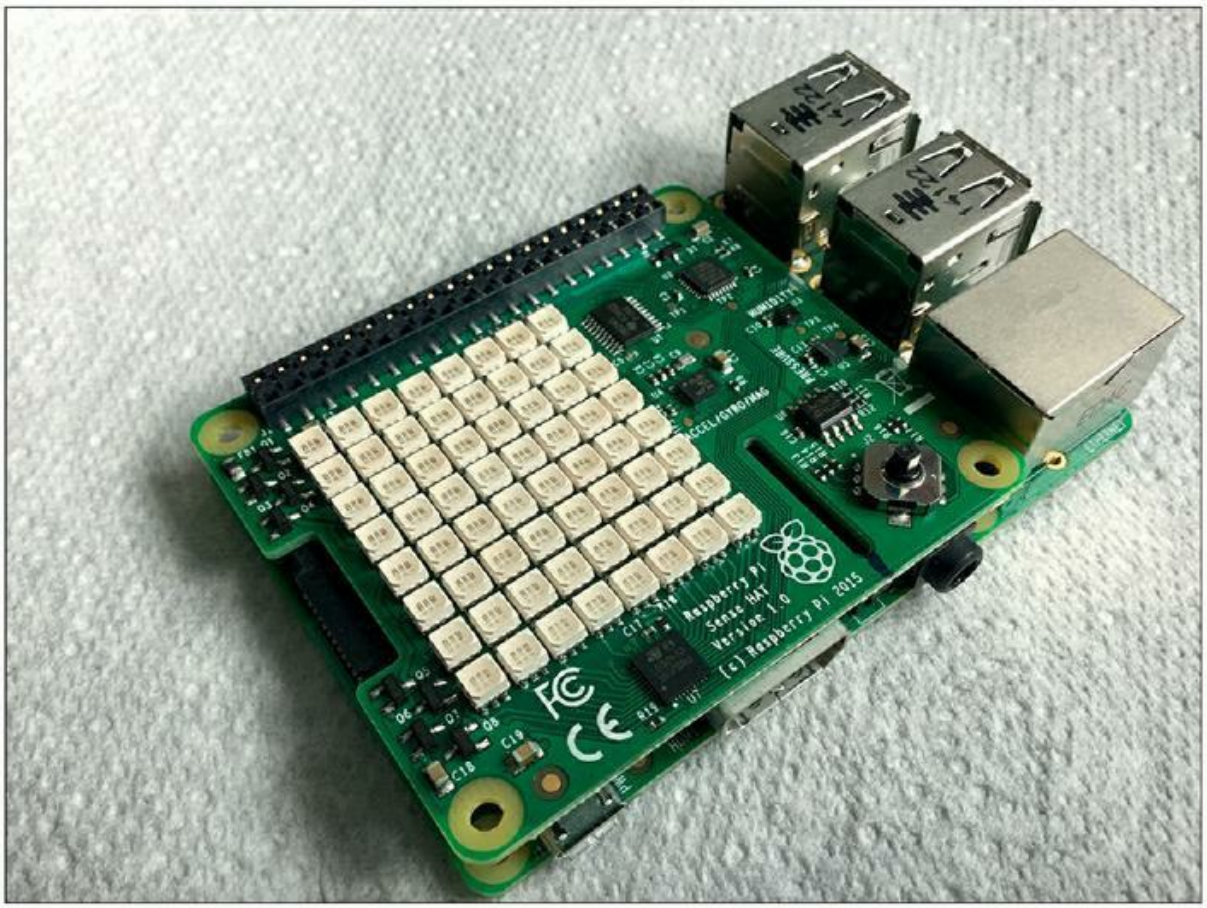


Figure 15.10 : La carte Sense HAT.

La carte Skywriter HAT

La carte Skywriter HAT (Figure 15.11) est un détecteur de champ électrique proche en 3D. Il suffit de faire des mouvements avec vos mains non loin de la carte pour qu'ils soient détectés par les fonctions de la librairie, ce qui permet de déclencher vos propres fonctions. Les gestes reconnus sont par exemple les balayages vers la gauche, la droite, le haut et le bas et les mouvements circulaires avec un doigt. La carte reconnaît également les contacts directs sur la surface. La distance d'action est d'environ 5 cm. La carte peut être montée derrière un écran transparent non conducteur tel qu'une plaque en acrylique.

Figure 15.11 : La carte Skywriter HAT.

La carte Xtrinsic Sense Board

Cette carte bon marché est équipée de capteurs, un peu comme la carte Sense HAT, mais sans la matrice de LED. Elle est fabriquée en partenariat avec le distributeur et fabricant de cartes Raspberry Pi nommé Farnell. Elle réunit un capteur de pression de haute précision, fonctionnant dans la plage de 50 à 110 kPa, un accéléromètre numérique trois axes et un magnétomètre 3D. Comme vous pouvez le voir dans la figure, elle occupe moins de la moitié de la surface de la carte RasPi. Elle est souvent utilisée avec des robots et des ballons stratosphériques.

Figure 15.12 : La carte Xtrinsic Sense Board.

Autres cartes

De nombreuses cartes d'extension sont disponibles sur le marché, certaines provenant de toutes petites entreprises, et d'autres qui sont à construire vous-même grâce à des descriptions disponibles sur le Web. Pour une première idée du vaste choix disponible, visitez la page suivante :

http://elinux.org/RPi_Expansion_Boards

Chapitre 16

Contrôler les circuits du RasPi

DANS CE CHAPITRE :

- **Les broches GPIO à utiliser**
- **Contrôler les broches GPIO en Scratch et en Python**
- **Faire clignoter une LED et détecter un poussoir**
- **Construire un dé électronique**
- **Construire une maquette de passage piéton**

Nous avons vu dans le chapitre précédent à quoi ressemblait le connecteur GPIO. Nous pouvons maintenant découvrir comment exploiter les broches de ce connecteur avec des logiciels. Nous allons voir comment procéder d'abord avec le langage Scratch, puis avec Python, beaucoup plus puissant en ce qui concerne le contrôle des entrées et des sorties.

Accès aux broches GPIO

Le connecteur GPIO est la porte d'accès au monde extérieur pour votre carte Raspberry Pi. Les broches peuvent servir en sortie, par exemple pour contrôler une diode LED, mais aussi en entrée pour lire l'état d'un bouton ou d'un capteur. Les broches sont disposées sous forme de deux rangées parallèles de 20 broches, parmi lesquelles les broches d'alimentation et de masse. Il existe plusieurs techniques pour se brancher sur ces broches, les moins coûteuses n'étant pas les plus pratiques. Il faut d'abord découvrir à quel signal correspond chaque broche. La Figure 16.1 montre l'affectation des broches du connecteur GPIO.

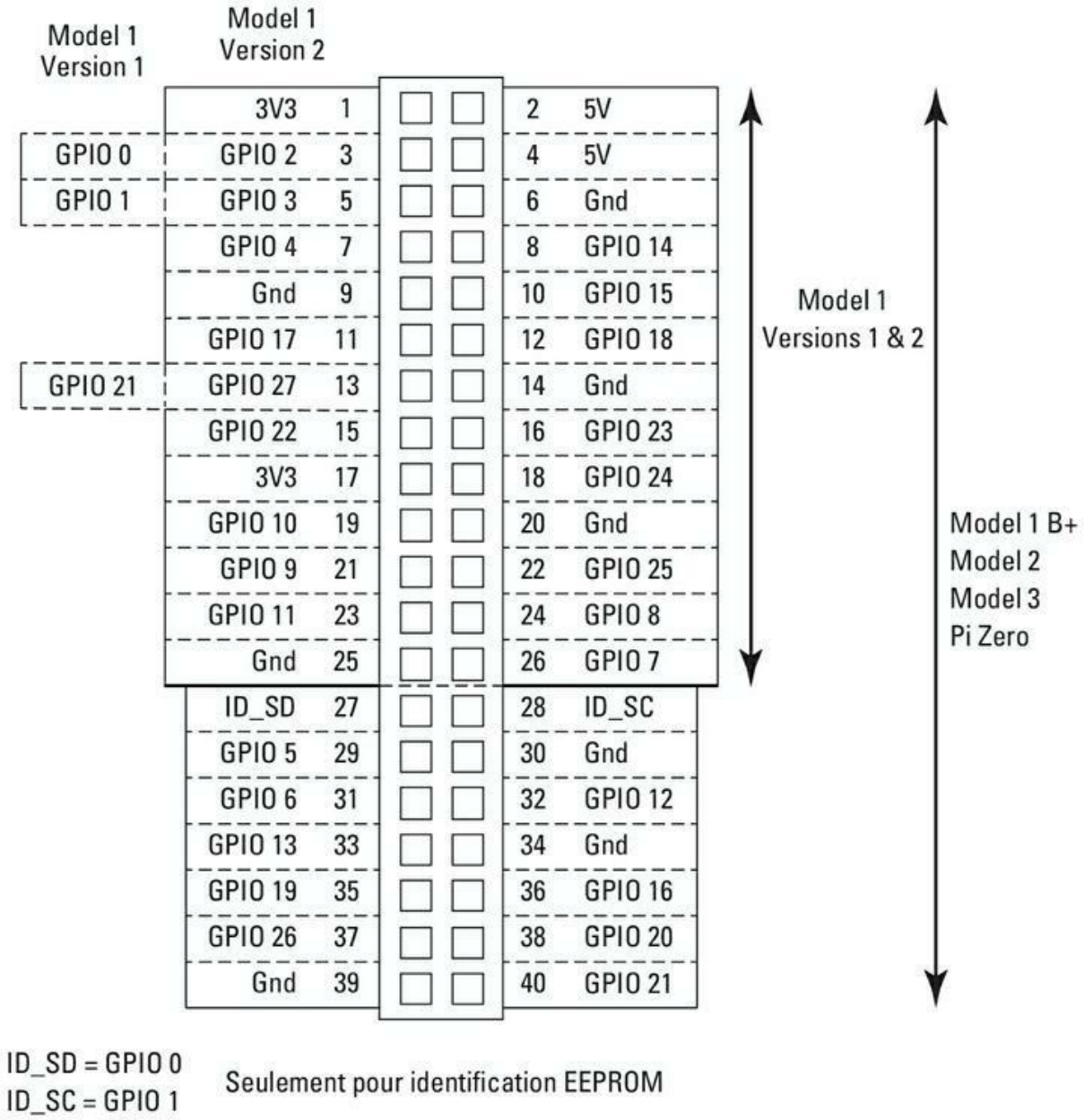


Figure 16.1 : Affectation des broches du connecteur GPIO.

Il existait au départ deux conventions pour désigner les broches. Dans la Figure 16.1, nous utilisons la convention BCM qui repère les broches telles qu'elles sont numérotées dans les fiches techniques décrivant les pattes de sortie du microcontrôleur Broadcom (d'où BCM). L'autre convention, qui n'est plus utilisée, s'appelait la convention Board. Les noms des broches étaient choisis en fonction des fonctions principales qui pouvaient être réalisées grâce à ces broches. L'idée était de rendre indépendant le brochage sur la carte RasPi des éventuels changements de brochage du microcontrôleur.

En réalité, seules les premières éditions de la carte RasPi ont eu à subir un changement dans l'affectation des broches. Et cela ne concernait que trois broches, visibles en haut à gauche dans la figure. Le modèle 1 avant le B+ n'offrait que 26 broches sur le connecteur, mais tous les modèles suivants

ont eu droit aux 40 broches qui sont devenues la norme. Vous n'êtes pas concerné, car vous aurez bien du mal à trouver un de ces anciens circuits. Nous donnons cette information au cas où vous auriez un de ces ancêtres dans un tiroir.

Lorsque l'on regarde le connecteur, on comprend qu'il n'est pas toujours facile de bien se repérer pour trouver un contact parmi les 40 sans se tromper. Une solution consiste à ajouter un gabarit avec des légendes, comme celui de la Figure 16.2.

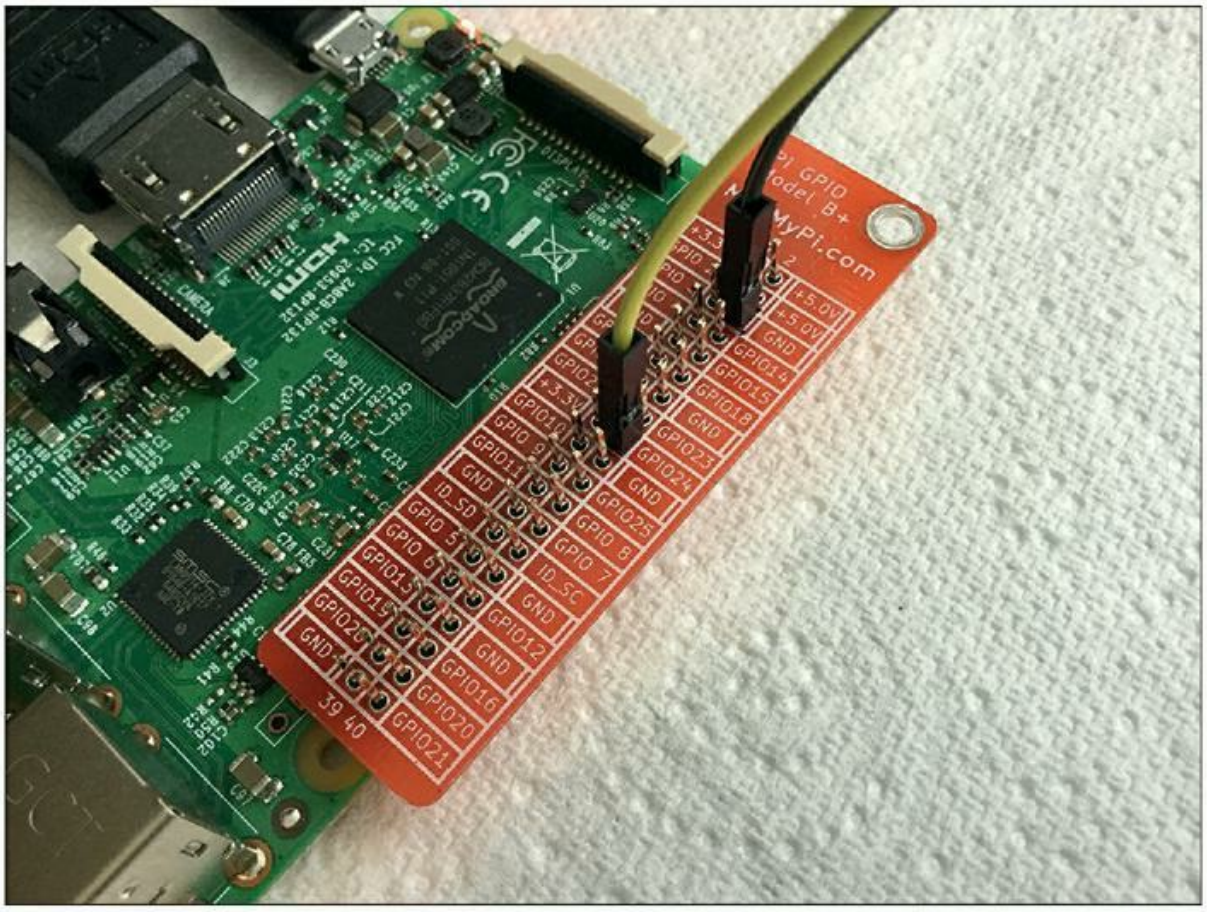


Figure 16.2 : Gabarit de repérage des broches GPIO.

Dans cette figure, c'est un gabarit en matériau dur, qui vient se poser sur les broches, sans les empêcher d'être utilisées. Nous vous conseillons fortement ce genre de gabarit de repérage si vous comptez effectuer les branchements fil à fil. Vous pouvez également imprimer un gabarit de légende puis enfoncer la feuille de papier avec patience sur les différentes broches. Une solution intermédiaire consiste à coller la légende sur un morceau de carton puis à percer des trous pour le faire passer sur les broches.

N'utilisez évidemment aucun matériau conducteur tel que de l'aluminium pour réaliser votre gabarit.

Soudage du connecteur sur le Pi Zéro

Les minicartes Pi Zéro offrent les mêmes 40 sorties GPIO que les autres modèles, mais le connecteur n'est pas soudé. Il faut donc réaliser cette soudure, ce qui offre certains avantages : vous pouvez choisir de mettre en place un connecteur femelle au lieu de mâle si cela convient plus à vos besoins. Vous pouvez même souder directement des fils volants dans les trous, ce qui convient dans le cas d'un montage qui doit devenir définitif.

Ne tentez jamais d'effectuer un branchement avec un fil volant en le faisant passer par un des trous. Vous êtes certain de créer des courts-circuits et d'endommager votre carte.

Tout le monde ne se sent pas à l'aise avec un fer à souder. La société Pimoroni a pensé aux ferrophobes en proposant un connecteur qui se met en place avec un marteau. Voyez la page suivante :

https://shop.pimoroni.com/products/gpio-hammer-header

Pour répondre à une forte demande, il existe dorénavant une variante de la minicarte Pi Zéro sur laquelle le connecteur est déjà soudé. Sa désignation est Pi Zéro WH.

L'opération de soudure de ce connecteur n'est pas si complexe, à condition de disposer d'un fer à souder avec une panne pointue. Nous avons vu les grands principes de la soudure dans le Chapitre 15. Voici quelques conseils supplémentaires. Vous commencez par insérer les broches mâles dans les trous puis vous retournez la carte pour qu'elle repose sur les broches. Calez le montage pour qu'il soit bien horizontal et stable. Servez-vous par exemple de pâte à modeler pour caler. Amenez le fer à souder à l'endroit où une broche touche le circuit imprimé et ajoutez un tout petit peu de soudure. Vous devez voir la soudure couler tout autour du trou. Attendez environ une demi-seconde que la soudure soit avalée par le trou par effet de capillarité. À ce moment, retirez vivement la panne. Ne faites pas bouger le montage. Si la soudure n'a pas été aspirée, soit il y en a trop, soit le fer n'était pas assez chaud.

Avant de passer à la broche suivante, vérifiez que le connecteur est absolument d'équerre par rapport à la carte. Si ce n'est pas le cas, faites une retouche en faisant fondre la première soudure. Important : dès que vous aurez soudé plusieurs broches, vous ne pouvez plus rattraper un connecteur qui a été engagé de travers. Réalisez ensuite toutes les autres soudures.

Si vous avez des soucis pour les soudures, il est possible que ce soit lié au fait que vous n'attendez pas suffisamment entre deux soudures. Il faut laisser au fer le temps de remonter en température. Nettoyer le faire sur votre éponge humide après chaque opération pour supprimer la résine.

Les explications ci-dessus peuvent sembler complexes, mais les choses sont beaucoup plus faciles dans la pratique. La Figure 16.3 montre une soudure en cours de réalisation.

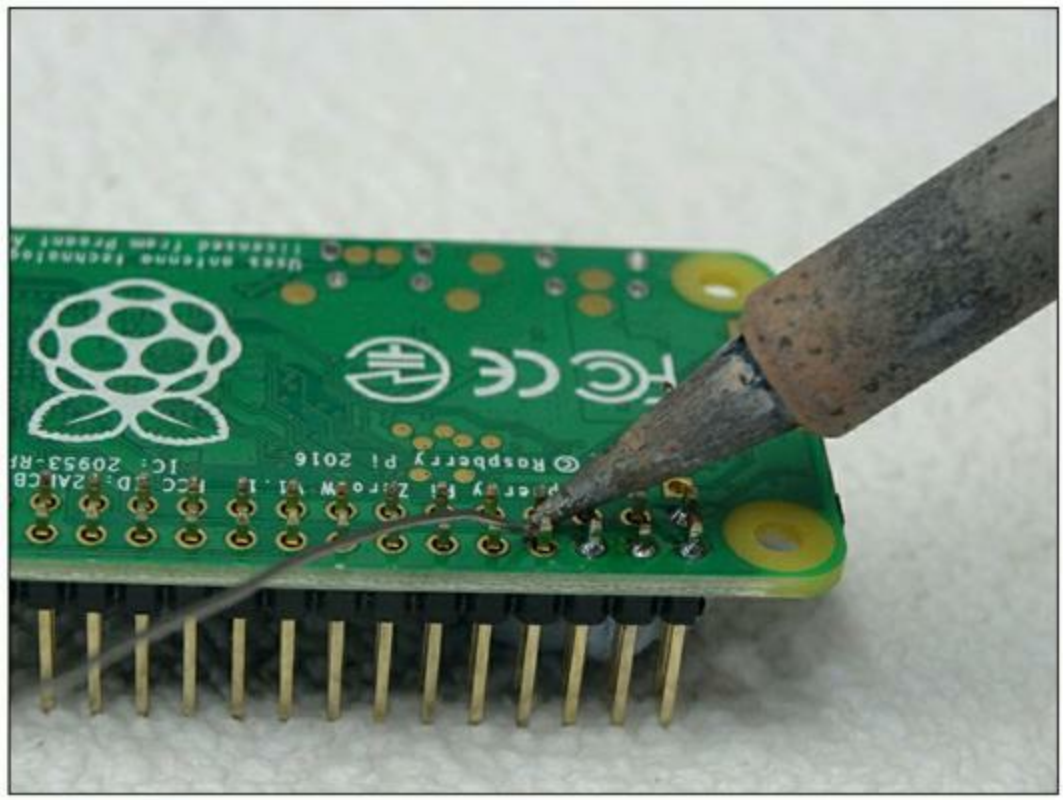


Figure 16.3 : Soudure d'un connecteur sur une carte Pi Zéro.

Les connecteurs d'extension

L'ajout d'un gabarit permet de brancher plus facilement les différents signaux sans se tromper, mais il est une solution encore plus efficace, consistant à déporter tous les signaux vers une plaque d'expérimentation sans soudure. Ce genre de plaque est omniprésent dans la réalisation de prototypes et de circuits d'essai. Une solution pour récupérer toutes les broches du connecteur GPIO sur une telle plaque consiste à adopter un circuit *Cobbler*, c'est-à-dire un câble en nappe combiné à un petit circuit imprimé.

La Figure 16.4 montre un exemple de prolongateur, le T-Cobbler Plus d'Adafruit.

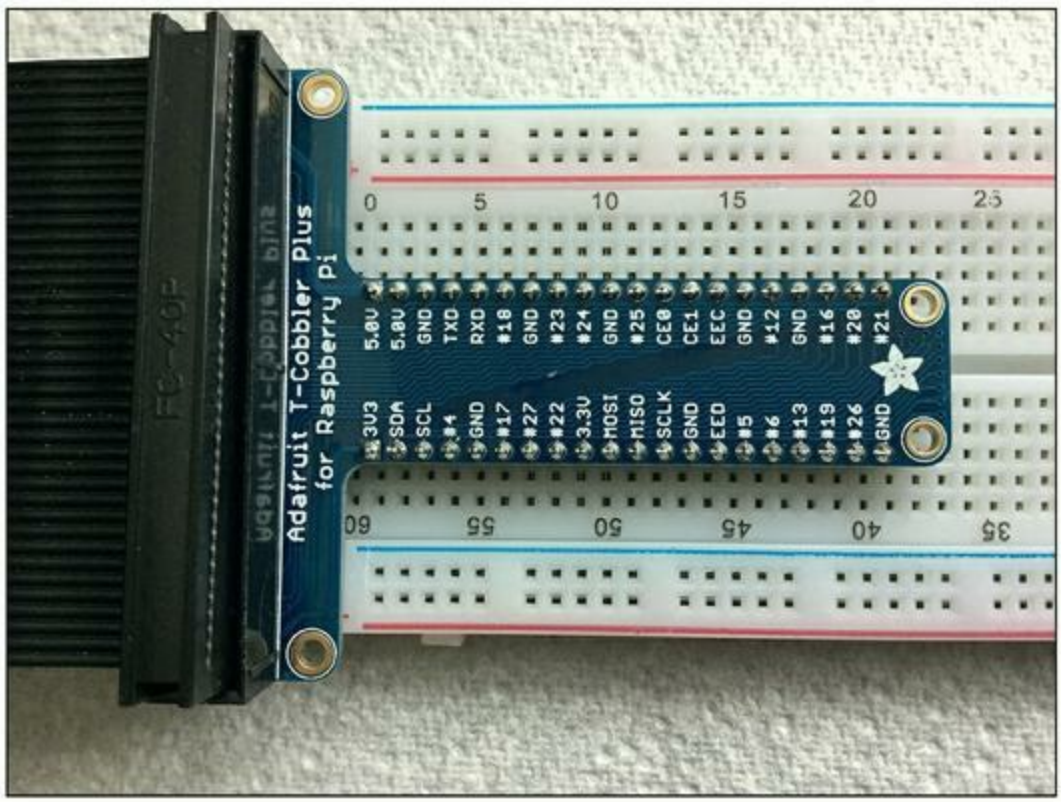


Figure 16.4 : Exemple de prolongateur de connecteur GPIO.

Un de nos prolongateurs préférés est celui de Dtronixs (www.dtronixs.com). Il comporte un petit morceau de circuit d'expérimentation qui vient se poser directement sur les broches du GPIO. Il comporte en complément trois diodes LED et deux boutons-poussoirs sur le côté (Figure 16.5).

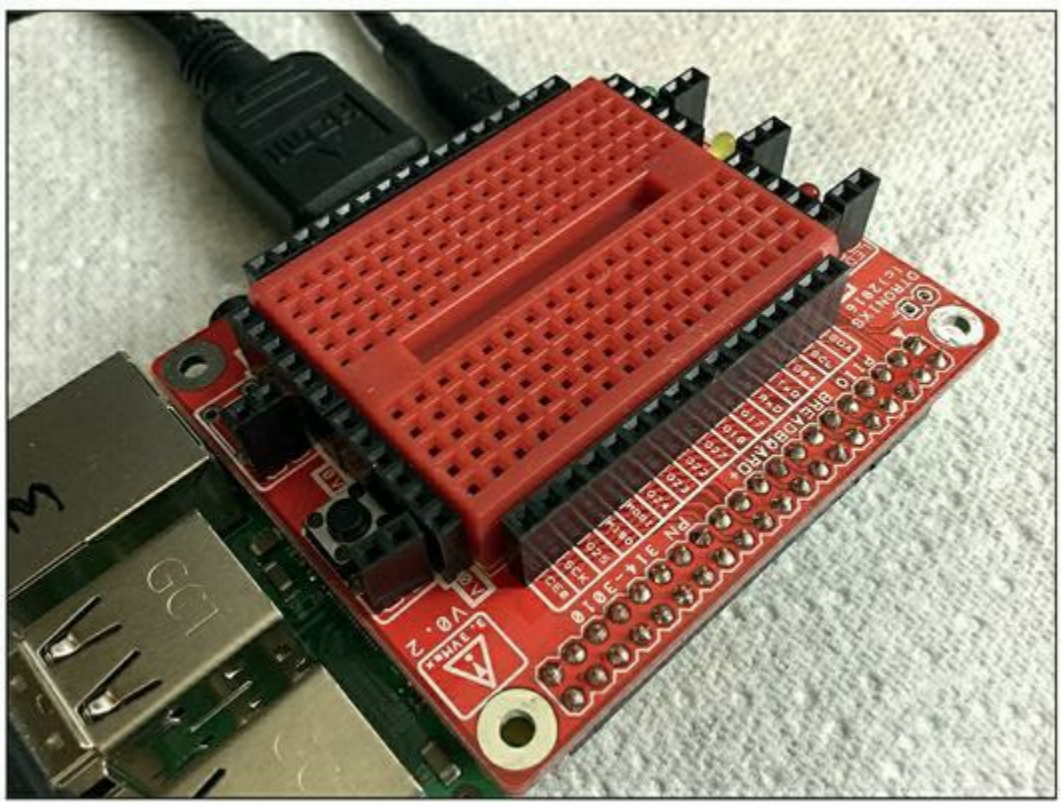


Figure 16.5 : Le prolongateur et la plaque d'expérimentation Dtronixs.

Réaliser les connexions

Une fois que vous savez à quelle broche vous brancher, il ne reste plus qu'à effectuer le branchement avec les autres composants. La solution la plus simple consiste à utiliser des fils volants que l'on appelle *straps* (voir en Figure 16.11). Un strap est un fil souple de 5, 10 ou 20 cm de long dont chaque extrémité est munie d'une broche dans un petit boîtier. La connexion entre le fil et la broche n'est pas réalisée par soudure, mais par sertissage. Cette technique est plus fiable que la soudure parce qu'il n'y a pas de point faible à l'endroit où se termine la partie soudée. En effet, c'est à cet endroit que le fil finit par casser au bout d'un certain nombre de pliages. À l'intérieur du boîtier, le connecteur possède une forme adaptée pour qu'un outil de sertissage puisse bien serrer le fil et le connecteur de manière solide.

Le connecteur GPIO du Raspberry Pi est un connecteur de type Dupont. L'embout d'un strap qui vient s'enficher sur une broche mâle est un connecteur femelle, et l'embout qui possède une broche pouvant s'enfoncer dans une plaque d'expérimentation est un connecteur mâle. Pour connecter une broche du connecteur GPIO vers un trou d'une plaque d'expérimentation, il faut donc un strap mâle vers femelle. Ce genre de strap est en général vendu sous forme de nappe multicolore, et vous séparez facilement un fil des autres en tirant doucement. Vous pourriez préparer vous-même vos straps avec une pince à sertir, mais il est bien plus pratique de les acheter déjà préparés. Pour les connexions entre deux trous d'une plaque d'expérimentation, il vous faut des straps mâle vers mâle. Vous pouvez aussi utiliser directement un fil dénudé et étamé sur l'extrémité. N'enfoncez pas trop vos fils dans ce cas. N'utilisez pas de straps trop longs, car cela va vous gêner pour mettre en place les autres composants. N'utilisez pas non plus de fil de trop gros diamètre. Nous conseillons de prendre du fil de câblage standard, également appelé SWG24 ou 24AWG. En Europe, cela correspond à du fil d'environ 0,25 mm.

L'électricité n'a aucune préférence au niveau de la couleur du fil dans lequel vous la faites circuler, mais les conventions internationales demandent d'utiliser du rouge ou de l'orange pour le plus de l'alimentation et du noir, du bleu ou du vert pour la masse ou pôle négatif.

Votre premier circuit

La première chose que doit essayer de faire toute personne qui découvre l'électronique avec une carte Raspberry Pi consiste à faire clignoter une diode LED. C'est l'équivalent électronique du message de bienvenue « Hello world » des débutants en programmation. Pour contrôler une diode LED, il vous faut la diode et une résistance pour limiter le courant qui va la traverser (afin de protéger la broche de sortie du GPIO). La Figure 16.6 montre le câblage de ce premier montage, avec un schéma de principe sur la droite.

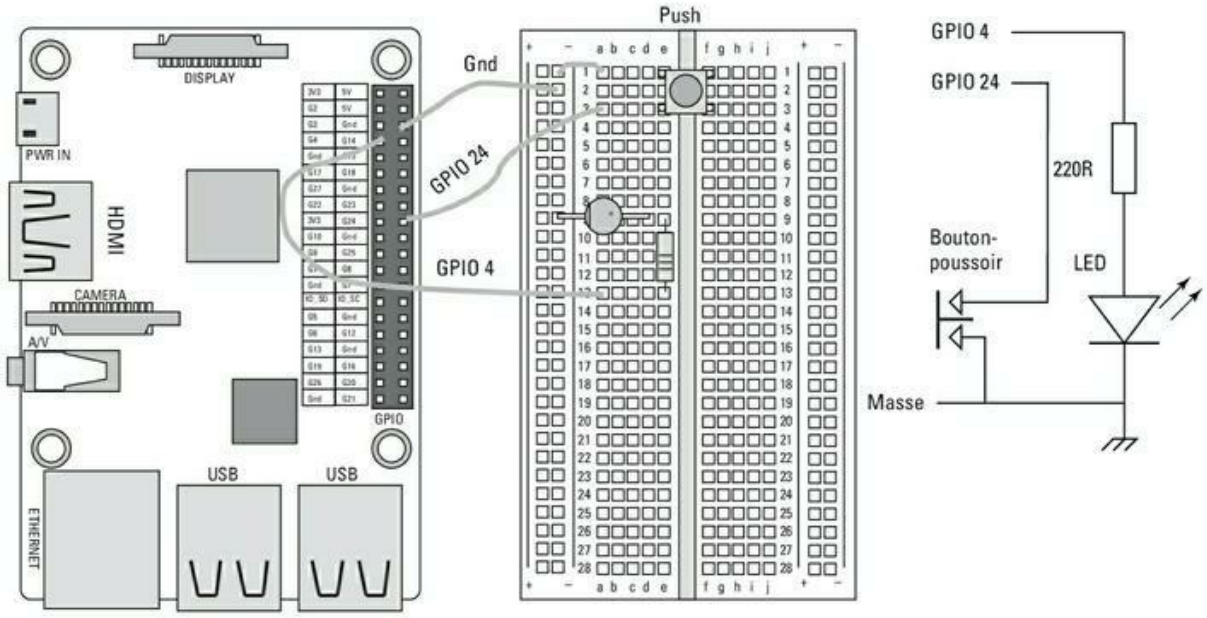


Figure 16.6 : Câblage d'une diode LED et d'un interrupteur sur deux broches GPIO.

Nous avons ajouté un bouton-poussoir qui nous servira à modifier le comportement de la diode LED.

Les deux points sur lesquels il faut être vigilant ici sont l'orientation du branchement de la diode LED et l'orientation du bouton-poussoir.

Une diode LED ne fonctionne que dans un sens. Le côté positif doit être connecté à l'anode. Il correspond à la partie plate du triangle. La cathode correspond au trait qui barre le fil sur le symbole ; elle doit être reliée à la masse ou pôle négatif. Ce n'est que dans ce sens que la diode LED va briller. Mais comment reconnaître les deux extrémités de la diode ? Sur une diode neuve, la patte de l'anode est la plus longue des deux. Sur une diode de récupération dont les pattes ont été recoupées, il faut regarder attentivement le bord inférieur du boîtier de la diode. La partie un peu aplatie correspond à la cathode, c'est-à-dire au moins. C'est le côté gauche de la diode sur la plaque d'expérimentation de notre figure.

Le bouton-poussoir doit lui aussi être orienté correctement, mais pas pour la même raison. Il s'agit d'un interrupteur simple qui ne comporte donc que deux contacts. Le sens dans lequel le courant passe n'a aucune importance. Mais la plupart des petits interrupteurs prévus pour être directement enfichés sur une plaque d'expérimentation ont 2 fois 2 pattes, pour des raisons de stabilité mécanique. Deux contacts sont reliés ensemble de chaque côté. Observez bien sur la figure comment les contacts sortent du boîtier. Quand vous appuyez sur le bouton, cela établit le contact entre les deux pattes d'un côté, et de même pour l'autre côté. Il faut donc insérer l'interrupteur correctement. Si vous vous trompez, c'est comme s'il y avait contact en permanence. Si vous êtes vraiment perdu, branchez un fil à un contact et l'autre au contact diamétralement opposé. Cela fonctionnera à coup sûr.

Une règle d'or en électronique consiste à ne jamais brancher ou débrancher quoi que ce soit quand le circuit est sous tension. Cette règle s'applique d'autant plus à votre carte RasPi. En effet, un circuit en cours de rebranchement peut être placé dans des conditions qui peuvent l'endommager. Cette opération s'appelle le branchement à chaud, et elle ne peut être réalisée que dans un ordre spécial, lorsqu'elle est autorisée. Les connecteurs USB sont prévus pour se brancher à chaud. Leurs deux contacts d'alimentation et de masse sont plus longs que les contacts des signaux, ce qui garantit que le courant est établi en premier.

Faire clignoter la diode LED

Une fois que le circuit est monté, nous pouvons passer au logiciel pour lui donner vie. Vous pouvez démarrer votre carte RasPi. Si vous constatez qu'elle ne démarre pas, c'est que quelque chose a été mal branché. Dans ce cas, débranchez immédiatement et vérifiez tout. Nous allons voir d'abord comment faire clignoter la diode LED avec Scratch, puis avec le langage Python 3.

Avec Scratch 1.4

Nous avons vu en détail le langage Scratch dans le Chapitre 9. Vous savez que le principe est d'assembler des blocs de fonctions pour constituer un script. Le langage a beaucoup de succès auprès des enfants car il ne demande pas beaucoup de saisie au clavier. Pour communiquer avec les broches du connecteur GPIO, Scratch utilise le bloc d'envoi de messages, **Envoyer à tous**.

Démarrez donc Scratch, et commencez par déposer dans la zone de scripts un bloc **Quand drapeau vert pressé** depuis la catégorie Contrôle. Dans

la même catégorie, cherchez le bloc **Envoyer à tous** et déposez-le juste sous le précédent. Ouvrez la liste de ce bloc, choisissez **Nouveau/Edit** et saisissez exactement le nom `gpioserveron` puis validez par OK. Vous demandez ainsi à Scratch d'utiliser les broches GPIO et de se préparer à leur envoyer des messages.

Déposez un deuxième bloc de diffusion **Envoyer à tous**, choisissez **Nouveau/Edit** et saisissez cette fois-ci `config4out`. Vous demandez ainsi d'utiliser la broche 4 en tant que sortie, ce qui permet de contrôler par programme l'état électrique de cette broche. Nous voulons pouvoir faire basculer l'état de la broche du niveau bas (Low) au niveau haut (High). Puisque nous avons branché la diode LED dans le bon sens, lorsqu'il y a une tension haute, c'est-à-dire 3,3 V sur la broche, le courant va passer à travers la diode LED et la résistance, et la diode va s'allumer. Quand nous forçons la broche de sortie à l'état bas, soit 0 V, la diode va s'éteindre. Dans la même catégorie de blocs de contrôle, prenez un bloc de répétition **Répéter indéfiniment** et déposez-le sous le bloc précédent. À l'intérieur de la boucle, ajoutez un bloc **Envoyer à tous**, en fournissant comme variable le nom `gpio-4high`. Ajoutez ensuite un bloc **Attendre x secondes** et saisissez la valeur `0.3`. Faites une copie de ces deux blocs par clic-droit puis Dupliquer et changez le nom pour le second bloc **Envoyer à tous** pour qu'il indique `gpio4low`. Vous avez fini votre programme. Le script doit avoir l'aspect de la Figure 16.7. Vous pouvez tout de suite essayer le programme en cliquant le drapeau vert en haut à droite. Vous devriez voir la LED clignoter.

Vous changez la vitesse de clignotement en modifiant les valeurs dans les deux blocs d'attente. Vous pouvez bien sûr indiquer une valeur différente dans un bloc par rapport à l'autre.

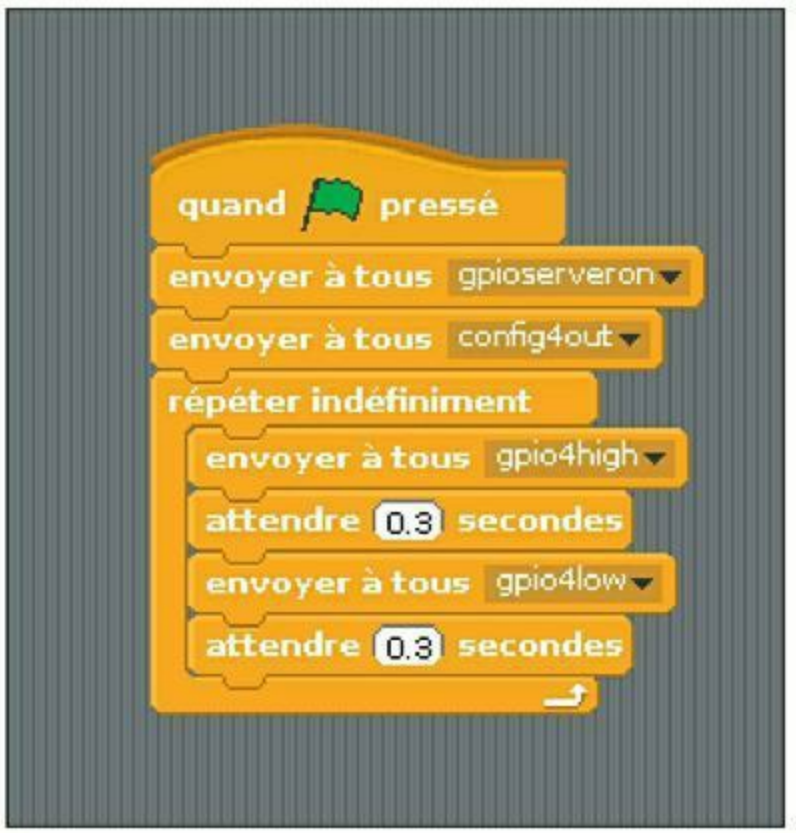


Figure 16.7 : Programme Scratch pour faire clignoter la diode LED.

Contrôler la vitesse de clignotement

Nous pouvons utiliser une broche GPIO en entrée de façon très simple : il suffit de placer un interrupteur entre la broche et la masse. Quand l'interrupteur est enfoncé, la broche est forcée à la masse. Quand le bouton n'est pas enfoncé, la broche n'est connectée à rien. Nous avons vu dans le Chapitre 15 que lorsqu'une broche n'est connectée à rien, l'ordinateur risque de lire une valeur totalement aléatoire, parce qu'il capte les moindres interférences qui traînent dans l'air. Pour que le montage fonctionne de manière fiable, il faut donc forcer l'entrée dans un état stable, en activant la résistance de tirage interne (*pull-up*). Cette résistance permet de relier l'entrée à une source d'alimentation équivalente à l'état haut. Autrement dit, lorsque rien n'est connecté à la broche, l'état sera haut. Ce sera donc le cas lorsque le bouton n'est pas enfoncé. Lorsqu'il est enfoncé, la broche sera forcée à l'état bas.

Lorsque vous ajoutez une entrée dans un script, vous devez en général ajouter un bloc conditionnel dans votre code. Selon l'état enfoncé ou non du bouton, nous allons pouvoir faire exécuter certaines instructions plutôt que d'autres. Nous allons voir dans l'exemple suivant comment choisir entre deux temps d'attente en fonction de l'état du bouton.

Vous pouvez repartir du précédent exemple ou pas. Il faut commencer par un bloc **Quand drapeau vert pressé** de la catégorie Contrôle puis un bloc de diffusion **Envoyer à tous** avec `gpioserveron`, suivi d'un autre bloc de diffusion avec `config4out`, comme dans le précédent exemple. Nous ajoutons un troisième bloc de diffusion, mais cette fois-ci avec le message `config24in` pour configurer la broche 24 en entrée, tout en activant sa résistance de tirage. Nous aurions pu indiquer l'un des trois messages `config24input`, `config24inpullup` ou `config24inputpullup` qui sont absolument synonymes, mais plus explicites pour indiquer que la résistance de tirage est mise en fonction.

Vous ajoutez ensuite une boucle de répétition, **Répéter indéfiniment**, dans laquelle vous placez un bloc conditionnel à deux branches **Si... sinon**. Ouvrez ensuite la catégorie Capteurs de Scratch et déposez un bloc **Valeur du capteur potentiomètre**. Ouvrez la liste déroulante pour choisir le message `gpio24`.

Si le nom `gpio24` n'est pas disponible dans la liste, lancez le programme avec le drapeau vert puis arrêtez-le avec le bouton rouge. Cela rafraîchit la liste des variables accessibles. Vous devriez ensuite pouvoir sélectionner ce nom dans la liste.

Accédez à la catégorie de blocs Opérateurs et déposez un bloc vert de comparaison ; il s'agit du petit bloc avec deux rectangles blancs séparés par un signe égal. Insérez le bloc **Valeur du capteur** dans le petit rectangle blanc de gauche et saisissez la valeur `1` dans le petit rectangle de droite. Déposez enfin l'ensemble dans l'emplacement prévu de la fourche **Si**.

Comme le montre la Figure 16.8, vous placez ensuite un bloc d'attente de 0,3 seconde dans la branche du **Si** puis un bloc d'attende d'une seconde dans la broche du **sinon**.

À la suite de ce bloc conditionnel, ajoutez un bloc **Envoyer à tous** avec le message `gpio4high`. Vous pouvez ensuite dupliquer tout le bloc conditionnel et rajouter la copie sous le précédent. Il ne reste plus qu'à ajouter un bloc de diffusion **Envoyer à tous** avec le message `gpio4low`. Vérifiez que votre programme correspond à la Figure 16.8.

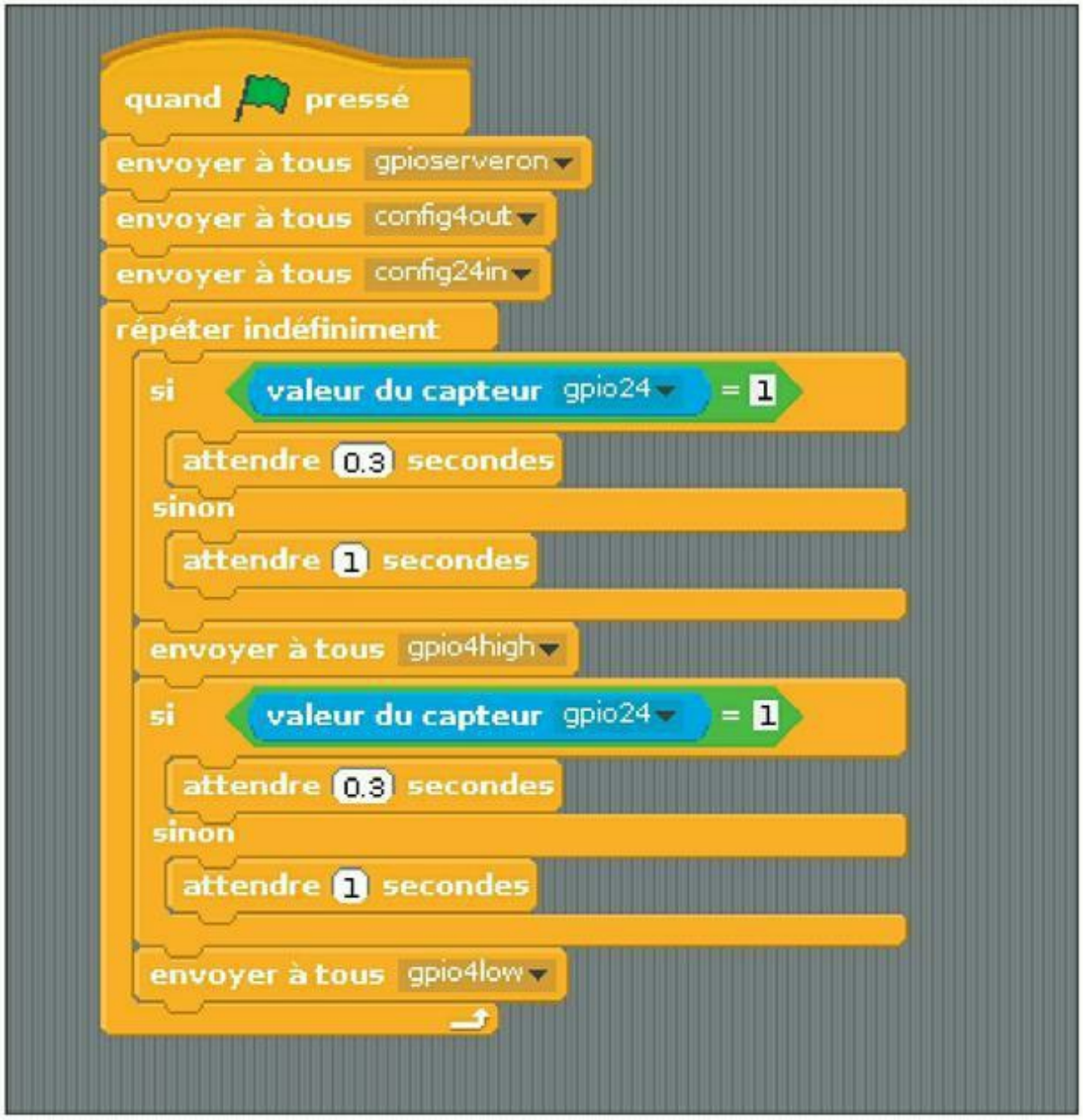


Figure 16.8 : Script Scratch pour contrôler une diode LED avec un bouton.

Lancez le programme pour voir clignoter la diode rapidement. Normalement, le clignotement ralentit lorsque vous enfoncez le bouton. Vous pouvez essayer de modifier le test du deuxième bloc conditionnel. Indiquez la valeur 0 à la place du 1. Avant de tester le résultat, devinez ce qui va se produire. Vérifiez ensuite si vous aviez bien deviné.

Scratch permet bien plus de choses que de contrôler des diodes LED et des boutons. Il est livré avec des fonctions pour contrôler des servomoteurs, des capteurs de distance, et même une caméra. Scratch est même doté de fonctions pour accéder à Internet et récupérer des bulletins météo. Le langage reconnaît aussi certaines cartes d'extension et de capteurs. Vous pouvez découvrir les autres utilisations du connecteur GPIO avec Scratch en visitant la page suivante :

www.raspberrypi.org/documentation/usage/scratch/gpi

Dans Scratch en version 2.0, les broches GPIO ne sont pas gérées par les mêmes blocs. Une incompatibilité majeure avec la version 1.4 est que les résistances de tirage interne des broches GPIO sont inversées : elles forcent à l'état bas et non à l'état haut. Dans ce cas, un bouton-poussoir doit être branché entre la broche GPIO et le positif, 3,3 V. Faites attention à ne pas vous brancher sur le 5 V, car cela pourrait endommager le RasPi.

Clignotement de LED avec Python

Nous avons découvert le langage Python dans le Chapitre 11. À la différence des blocs graphiques de Scratch, c'est un langage totalement textuel. Sa puissance et sa polyvalence proviennent en partie des centaines de librairies de fonctions complémentaires qui permettent d'enrichir le langage de base. Ces fonctions sont écrites en langage Python ou dans un autre langage de programmation.

Lorsqu'il s'agit d'exploiter les broches GPIO en Python, plusieurs librairies sont disponibles. Elles permettent un accès standard aux entrées et sorties, mais également à l'exploitation des fonctions spéciales de certaines broches. Nous avons choisi d'écrire les deux prochains exemples pour qu'ils correspondent le mieux possible aux deux exemples Scratch que nous venons de réaliser, ce qui vous permettra de comparer les deux langages.

Dans les premiers jours du Raspberry Pi, vous ne pouviez accéder aux broches GPIO qu'en démarrant le système dans le mode superutilisateur (super-user). Les choses ont heureusement changé ; vous pouvez dorénavant utiliser la plupart des librairies dans le mode utilisateur normal.

Le nouvel atelier de développement Thonny Python IDE est très apprécié des débutants, mais nous préférons de temps à autre continuer à utiliser l'atelier classique de Python qui porte le nom IDLE 3. Les deux ateliers sont accessibles depuis le menu Programmation de votre bureau. Choisissez celui que vous préférez pour créer un nouveau fichier puis saisissez toutes les instructions du Listing 16.1. Nous rappelons qu'en langage Python les majuscules ne sont pas confondues avec les minuscules. En outre, les espaces de marge gauche sont significatives. Dans l'exemple, il faut notamment deux espaces au début de chacune des quatre dernières lignes.

Listing 16.1 : Exemple de clignoteur en Python.

```
#!/usr/bin/python3
import time
import RPi.GPIO as io           # Comme RPi.GPIO

io.setmode(io.BCM)
io.setup(4,io.OUT)      # Broche en sortie

print("LED blinker - par Mike Cook")
print("Ctrl C pour quitter")
while True:
  io.output(4,0)
  time.sleep(0.30)
  io.output(4,1)
  time.sleep(0.30)
```

Les exemples sont disponibles sur le site de l'éditeur comme indiqué en introduction du livre. Repérez-vous au numéro de listing.

Enregistrez le fichier et lancez l'exécution. Vous devriez voir clignoter la diode LED comme dans le premier exemple Scratch.

Parcourons le Listing 16. 1 ligne par ligne. Après la première ligne technique, qui rappelle où se trouve l'exécutable de Python, nous arrivons à une directive **`import`** qui demande d'importer dans notre projet les fonctions qui sont définies dans la librairie nommée *time*. Certaines de ces fonctions vont nous servir pour ajouter une pause. Nous importons également le paquetage nommé *RPi. GPIO* qui se charge des accès aux broches du connecteur GPIO. La plupart des programmeurs choisissent comme alias, c'est-à-dire comme nom plus court, le mot GPIO ; nous préférons être encore plus brefs en choisissant `io`. Cela nous fait gagner du temps pendant la saisie et nous évite de passer en majuscules. Nous arrivons ensuite à une première fonction qui décide de la convention de numérotation des broches, c'est-à-dire ici le format BCM qui est devenu le standard.

L'instruction suivante déclare que nous allons utiliser la broche 4 en tant que sortie. Nous utilisons ici la méthode **`setup()`** qui attend deux paramètres d'entrée, le numéro de la broche, puis le mode de configuration. La mention out signifie sortie. En réalité, c'est une valeur numérique qu'il faut indiquer en deuxième paramètre, mais grâce à la librairie, nous utilisons une constante prédéfinie. Du fait qu'elle appartient

à l'objet que nous avons créé, nous ajoutons **io.** en préfixe. Nous arrivons ensuite à deux lignes d'affichage d'un message de bienvenue puis nous entrons dans une boucle perpétuelle de type **`while True :`** . Toutes les lignes qui seront répétées sont décalées de deux espaces.

Dans la boucle, nous commençons par forcer la broche 4 à zéro, ce qui éteint la diode LED. Puis nous entrons dans une pause de 300 ms, et allumons la diode LED en forçant la broche 4 à l'état haut. Nous terminons avec une dernière pause de la même durée. Il arrive souvent que l'on oublie cette dernière pause, ce qui fait que la diode semble rester éteinte, car elle reste allumée trop peu longtemps. Le contrôle d'une broche de sortie n'est donc pas si compliqué en Python.

Voyons maintenant comment traduire en Python le deuxième exemple, celui où nous contrôlons le clignotement grâce au bouton-poussoir. Cela correspond au Listing 16.2. Vous pouvez repartir du précédent exemple en l'enregistrant sous un nouveau nom puis en saisissant les lignes ci-après.

Listing 16.2 : Contrôle du clignotement par un bouton-poussoir.

```
#!/usr/bin/python3

io.setmode(io.BCM)
io.setup(4,io.OUT)
io.setup(24,io.IN, pull_up_down=io.PUD_UP)

print("LED blinker - par Mike Cook")
print("Ctrl C pour quitter")
while True:
  io.output(4,0)
  if io.input(24) == 1:
     time.sleep(0.30)
  else  :
     time.sleep(1.00)
  io.output(4,1)
  if io.input(24) == 1:
     time.sleep(0.30)
else  :
```

```
        time.sleep(1.00)
```

La nouveauté de ce programme est l'utilisation d'une broche GPIO en entrée. La commande correspondante attend trois paramètres : le numéro de la broche, une constante pour définir que la broche sert en entrée, et une commande pour activer la résistance de tirage interne. Dans la boucle perpétuelle, nous commençons par éteindre la diode LED puis nous lisons l'état du bouton-poussoir au moyen de la méthode **`input (24)`**. La méthode renvoie soit zéro, soit un en fonction de l'état de la broche. La valeur est comparée à un et nous ne marquons la pause courte que si les deux valeurs sont égales. Dans le cas contraire, nous faisons une pause d'une seconde. Nous allumons ensuite la diode LED puis nous revenons au début de la boucle.

La librairie GP Zéro

Une nouvelle librairie est récemment apparue pour simplifier l'utilisation du connecteur GPIO en langage Python : elle porte le nom GPIO Zéro. Il ne faut pas la confondre ni avec les minicartes Raspberry Pi Zéro, ni avec la librairie pour les jeux vidéo Pigame Zéro. La librairie GPIO Zéro a été créée selon les principes de la programmation orientée objet. Elle définit donc une classe à partir de laquelle vous pouvez créer des objets qui vont posséder des méthodes et des données membres. C'est très différent de l'approche classique consistant à définir des fonctions (comme c'est le cas de la librairie RPi. GPIO et de bien d'autres). Chaque broche devient un objet qui doit être créé et configuré avant d'être utilisé. Cette complication apparente rend ensuite la programmation beaucoup plus simple. Voici comment nous pourrions réécrire l'exemple de clignotement de diode LED du Listing 16.1 (Listing 16.3).

Listing 16.3 : Programme clignoteur avec GPIO Zéro.

```
#!/usr/bin/python3
import time, os
import gpiozero as io   # Librairie Zero

led = io.LED(4)         # Broche 4 en sortie

print("LED blinker avec gpiozero - par Mike
Cook")
```

```
print("Ctrl C pour quitter")
while True:
  led.on()
  time.sleep(0.30)
  led.off()
  time.sleep(0.30)
```

Cet exemple ressemble beaucoup à celui du Listing 16.1, mais c'est uniquement lié au fait que nous avons voulu qu'il lui ressemble. Avec cette librairie, vous utilisez obligatoirement la convention de numérotation BCM des broches. Il n'y a donc pas besoin de prévenir la librairie de quel système nous voulons utiliser. Une broche de sortie porte toujours le nom LED, qu'elle contrôle une diode LED, un moteur ou un autre composant. Pour configurer la broche en sortie, nous écrivons donc tout simplement **`io. LED(4)`**. Il s'agit d'un appel à une méthode, qui renvoie une valeur qui est une référence à un objet de cette classe. Nous verrons plus en détail les classes et les objets dans le prochain chapitre, car nous allons y définir une classe dans un exemple.

Pour l'instant, sachez que c'est une technique permettant d'utiliser le même bloc d'instructions pour gérer plusieurs objets, par exemple plusieurs broches GPIO. Chaque broche, donc chaque objet, doit être déclarée dans une instruction distincte. Cette opération permet au programme d'obtenir un identifiant numérique qui lui permet ensuite de gérer cet objet. Le code est stocké dans une variable que nous choisissons d'appeler `led` dans l'exemple. Cette variable est la référence de l'instance ou de l'objet. Dans la classe sont définies toute une série de méthodes, c'est-à-dire d'actions réalisables par l'objet. Pour appeler une méthode, il suffit de donner le nom de l'objet, suivi d'un point, suivi du nom de la méthode et de ses paramètres. Pour allumer ou éteindre une diode LED, il suffit donc d'écrire **`led.on()`** ou **`led.off()`**.

Cette simplification impose des limitations. Vous ne pouvez pas envoyer une valeur numérique dans une variable pour contrôler la diode LED. Vous devez spécifier le nom de la méthode qui a l'allumage ou l'extinction comme effet. Cette limitation est volontaire, car la librairie GPIO Zéro parvient ainsi à simplifier l'accès aux broches. Il ne s'agit pas d'une limitation de la programmation orientée objet. Il existe des techniques pour passer outre cette contrainte, mais le fait de les adopter rend le langage plus complexe que prévu au départ.

Pour bien comprendre à quel point cette librairie permet de simplifier les choses, il suffit de constater la taille du Listing 16.4 qui a exactement le même effet que le Listing 16.3.

Listing 16.4 : Clignotement d'une LED en orientation objet.

```
import gpiozero as

ioled = io.LED(4)
led.blink()
```

Il n'y a rien d'autre à écrire ! Cette simplification est très appréciée des débutants, mais elle ne permet pas d'apprendre grand-chose en programmation. Cela permet tout du moins de bien se concentrer sur les activités de montage des circuits électroniques. La deuxième instruction accepte des paramètres pour contrôler la vitesse de clignotement. Voici par exemple ce qu'il faut ajouter pour que le résultat soit strictement identique à l'exemple initial au niveau des durées :

```
led.blink(on_time = 0.3, off_time = 0.3)
```

La méthode **`blink()`** accepte même d'autres paramètres. Voici comment enrichir la même instruction pour forcer un fondu enchaîné vers l'allumage et vers l'extinction :

```
led.blink(on_time = 2.0, off_time = 2.0,
fade_in_time = 1.0, fade_out_time ↲
= 1.0 )
```

Ces paramètres sont très pratiques lorsque vous n'avez pas besoin d'autre chose, ce qui est idéal pour faire ses premiers pas.

Voyons maintenant comment écrire dans cette orientation objet l'exemple qui contrôle le clignotement grâce à un interrupteur ou bouton-poussoir. Le résultat est fourni dans le Listing 16.5.

Listing 16.5 : Clignoteur contrôlé par bouton avec GPIO.

```
#!/usr/bin/python3
import time, os
import gpiozero as io    # avec GPIO Zero

led = io.LED(4)          # Broche 4 en sortie
inter = io.Button(24)    # Broche 24 en entree
```

```
print("LED blinker avec gpiozero - par Mike
Cook")
print("Ctrl C pour quitter")
while True:
led.on()
if inter.is_pressed:
   time.sleep(0.30)
else  :
   time.sleep(1.00)
led.off()
if inter.is_pressed:
   time.sleep(0.30)
else  :
   time.sleep(1.00)
```

La préparation de la broche d'entrée ressemble beaucoup à celle de la broche de sortie, sauf que nous demandons de créer un objet nommé `inter` (*push*) à partir de la classe nommée **`Button`**. Nous nous servons ensuite de la méthode **`is_pres-sed`** avec le nom d'objet en préfixe pour définir les délais de clignotement. Le simple fait de créer un objet à partir de cette classe active la résistance de tirage interne.

Cette classe d'entrée offre de nombreuses options. Vous pouvez notamment définir une durée d'annulation du rebond, c'est-à-dire le temps qu'il faut attendre avant de prendre en compte l'état du bouton. Nous reparlons du rebond plus loin dans ce chapitre. Les autres méthodes de cette classe permettent de savoir si le bouton est maintenu enfoncé (**`is_held`**), de se placer en attente d'un appui avec **`wait_for_press`**, en attente d'un relâchement avec **`wait_for_release`**, ou encore de déclencher une action si le bouton est enfoncé ou relâché. Ces deux dernières fonctions provoquent chacune le lancement d'une fonction externe qui va s'exécuter dans un autre programme, plus exactement un autre fil d'exécution que l'on appelle un exétron. Le système va alterner entre le programme et l'exétron jusqu'à la fin d'exécution de la fonction. Ce mécanisme permet à un programmeur, même débutant, de profiter d'une technique complexe basée sur les fonctions de rappel (*callback*). (Il faudra cependant apprendre à les maîtriser un peu plus pour savoir mettre au point un programme rétif.) Pour l'instant, regardons comment nous pouvons reformuler le code de clignotement avec un bouton grâce à deux fonctions de rappel externes (Listing 16.6).

Listing 16.6 : Clignoteur contrôlé par deux fonctions de rappel.

```
#!/usr/bin/python3
import gpiozero as io
from signal import pause

led   = io.LED(4)
inter = io.Button(24)

def clignoVite():
    led.blink(on_time = 0.3, off_time = 0.3)

def clignoLent():
   led.blink(on_time = 1.0, off_time = 1.0)

inter.when_pressed  =  clignoVite
inter.when_released =  clignoLent

pause()
```

La fonction **`pause()`** qui se trouve tout à la fin du programme va effectivement provoquer l'arrêt de son exécution. Les seules actions qui se produiront à partir de ce moment seront les appels aux deux fonctions de rappel selon que le système a détecté que le bouton a été enfoncé ou relâché.

Pour profiter au mieux de la librairie GPIO Zéro, vous devrez plonger dans la documentation en ligne. Avec la librairie GPIO, vous obtenez rapidement des résultats, sans avoir à beaucoup réfléchir, ce qui est déjà parfaitement convenable. Cela dit, vous n'aurez pas beaucoup appris en termes de programmation générale, justement à cause de cette simplification. Dès que vous aurez envie de créer des projets électroniques plus originaux, vous constaterez que la librairie GPIO Zéro est trop limitée.

Construire un dé électronique

Puisque nous savons maintenant comment exploiter les broches GPIO, nous pouvons passer à la création d'un vrai projet. Comme premier

exemple, essayons de construire un dé électronique.

Alea jacta est !

Cette phrase latine signifie « Les dés sont jetés ! ». Effectivement, en latin un dé se dit *alea*, et c'est le nom que nous allons utiliser comme identifiant pour une de nos variables.

Un dé électronique constitue un bon projet pour démarrer parce que vous allez pouvoir vous en servir avec vos jeux de société, et parce qu'il permet de présenter quelques concepts de programmation fondamentaux. En termes matériels, il s'agit de sept diodes LED avec leur résistance et d'un bouton-poussoir. Nous ne nous écartons donc que très peu des exemples du début du chapitre. En revanche, il est essentiel de bien positionner les diodes pour que le résultat ressemble à une face de dé. Voyons d'abord en Figure 16.9 le schéma de principe.

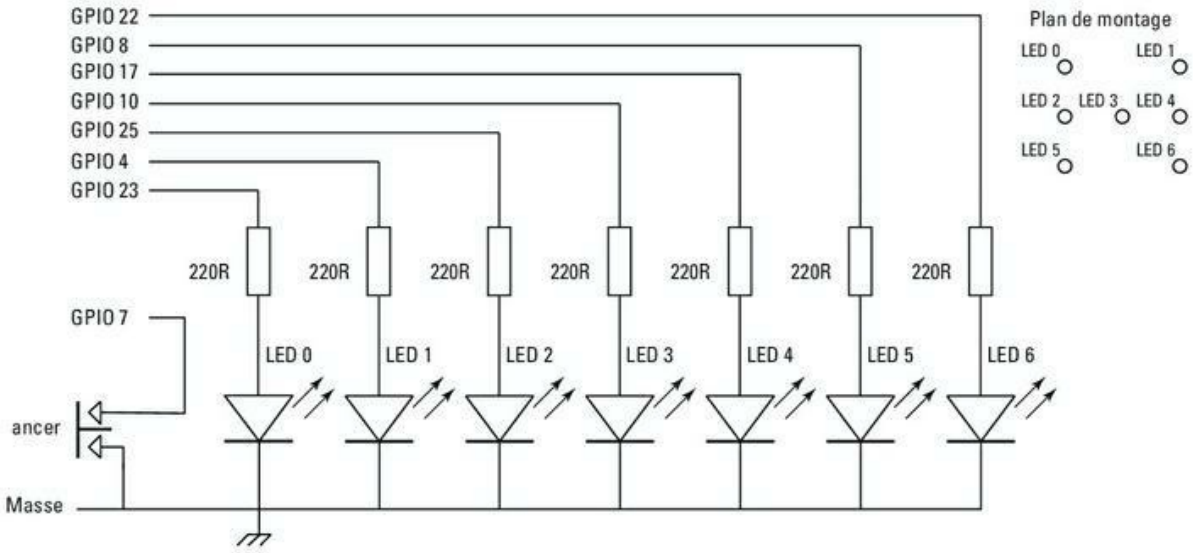


Figure 16.9 : Schéma de principe du dé à diodes LED.

Les numéros des broches GPIO que nous utilisons ne sont pas figés. Si vous vous trompez dans le câblage, il suffira de modifier le numéro dans le programme. Pour vous simplifier la vie, essayez de vous en tenir aux numéros de broches que nous suggérons. Le schéma est assez simple à lire. Il obéit aux conventions habituelles : le plus de l'alimentation est en haut et la masse en bas. Il n'y a ici aucun croisement de fil. Le courant circule de la gauche vers la droite. Ce schéma une fois traduit en implantation physique donne le résultat de la Figure 16.10. Nous utilisons ici à droite une plaque d'expérimentation sans soudure.

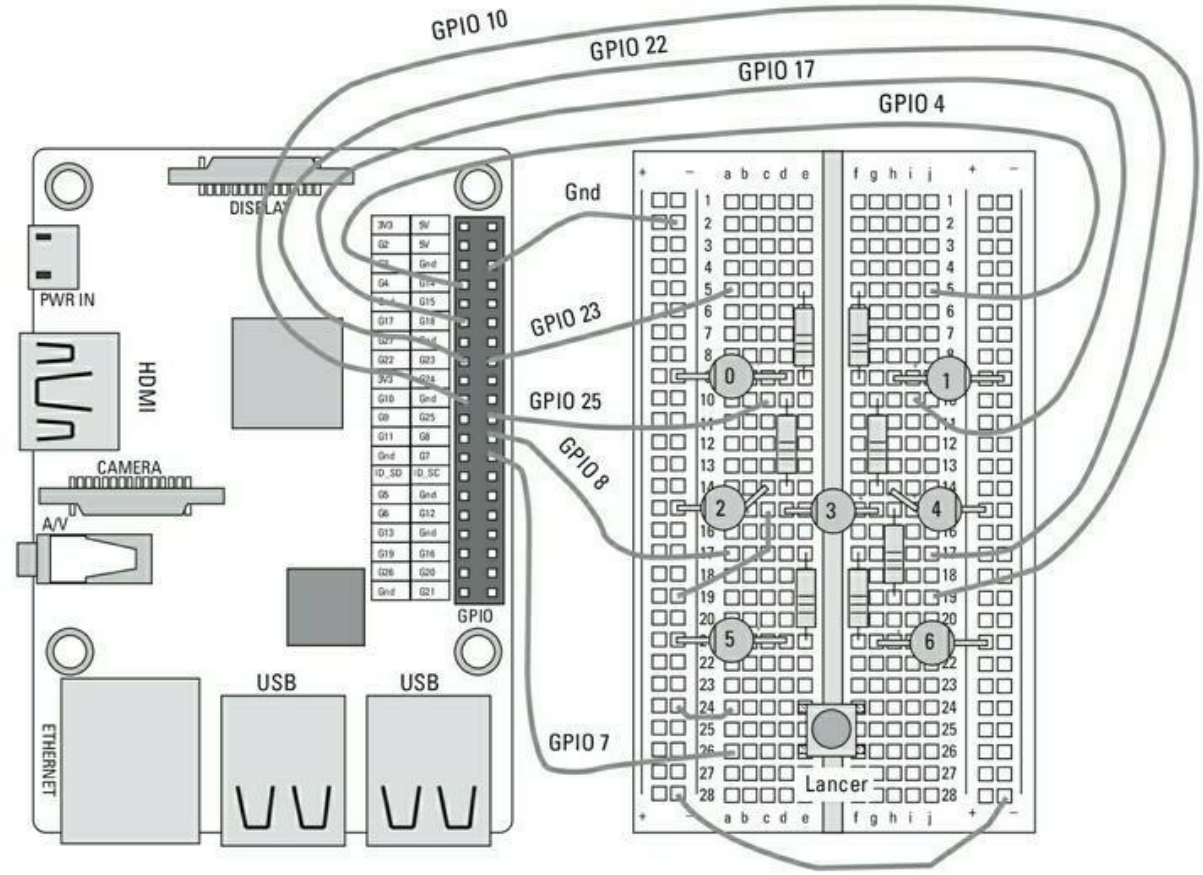


Figure 16.10 : Plan de câblage du dé électronique.

Ce plan devient un peu moins facile à lire, mais il montre la position physique exacte de chacun des composants. Regardez bien dans quels trous de la plaque sont enfoncés chaque patte de composants et chaque strap. Une patte de chacune des LED 2 et 4 a été un peu désaxée pour que ces LED restent en ligne avec la LED 3. Il est en général facile de passer du schéma de principe au plan de câblage, alors que l'inverse est toujours difficile. Dans notre plan, nous nous sommes simplifié la vie en choisissant les numéros des broches de manière à éviter de faire se croiser des fils. C'est la raison pour laquelle les numéros des broches semblent ne pas se suivre. En évitant tout croisement, nous rendons le plan un peu plus lisible. Notez que c'est rarement possible en réalité.

Nous vous conseillons de ne pas vous habituer à dépendre d'un plan de câblage. Prenez l'habitude d'apprendre à traduire de tête un schéma de principe en plan. Il n'est pas non plus nécessaire de préparer tout le plan de câblage avant de commencer à câbler. Travaillez fil par fil et construisez progressivement. C'est une compétence facile à acquérir qui vous rendra bien des services lors de vos futurs projets.

La Figure 16.11 montre l'aspect visuel du projet câblé. Nous avons choisi d'utiliser des diodes LED de 3 mm de diamètre. Vous constatez que le résultat visuel est encore plus fouillé que le plan de câblage. Il y a

plusieurs raisons à cela : on ne peut pas faire cheminer les fils volants exactement comme on le voudrait. De plus, le but d'une plaque d'essai est de permettre de réutiliser les composants. De ce fait, vous ne coupez pas les pattes à ras, et laissez donc les composants à une certaine hauteur par rapport à la plaque pour garder aux pattes toute leur longueur.

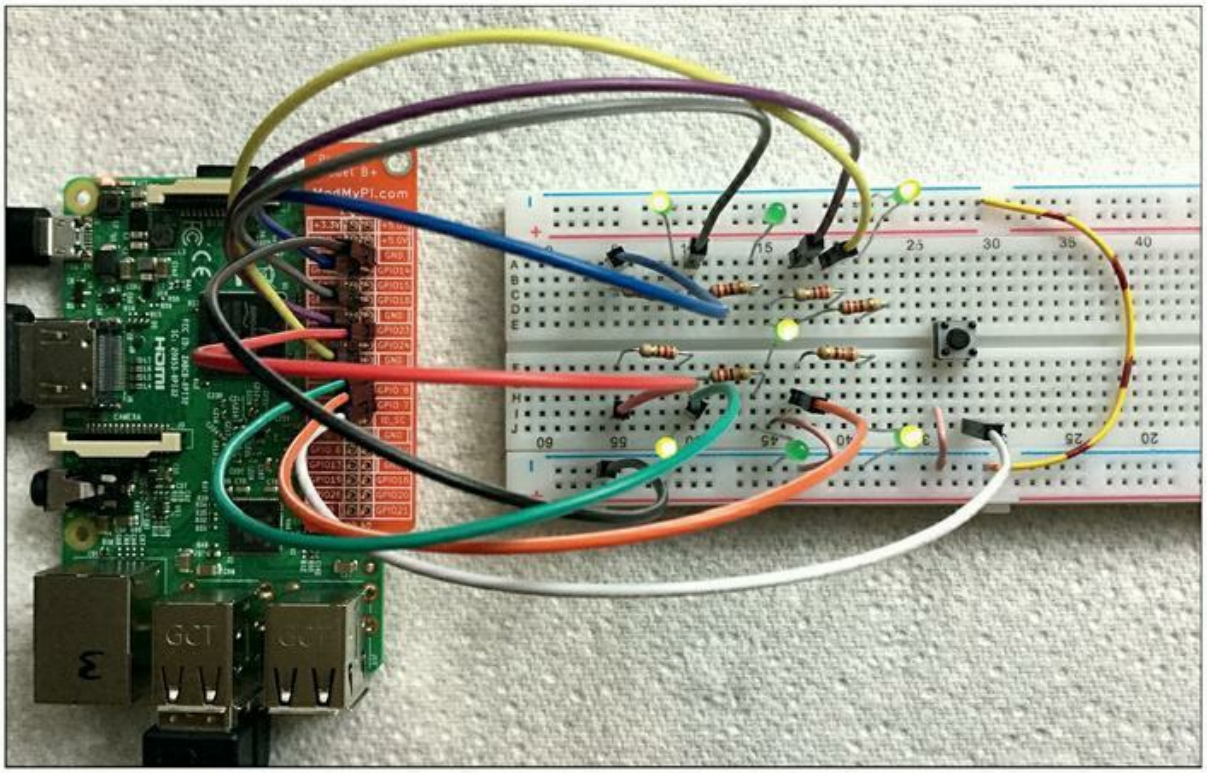


Figure 16.11 : Photographie du projet de dé électronique.

Test du câblage

Quelle que soit l'importance du projet, il est toujours conseillé de progresser étape après étape et de faire des tests intermédiaires. Vous devez être certain que le câblage est correct avant d'envisager la programmation. Vous pouvez par exemple partir du montage du clignotement d'une seule diode LED ou bien augmenter le programme de clignotement pour qu'il fasse clignoter les sept diodes LED. Même dans la version avec une seule LED, vous pouvez les tester toutes en amenant le fil de chaque LED tour à tour sur la broche qui contrôle le clignotement. Pensez à arrêter l'exécution du programme entre deux recâblages. Faites ceci jusqu'à ce que toutes les LED aient été testées, sans oublier le bouton-poussoir.

La partie programmation du projet comporte deux blocs principaux : la génération du nombre entre un et six et l'affichage de ce nombre.

Générer le nombre

Il y a deux techniques pour produire un nombre à peu près au hasard : soit utiliser le générateur aléatoire du contrôleur, soit détecter un événement imprévisible. Le problème du générateur aléatoire est qu'il n'est pas vraiment aléatoire. C'est un générateur pseudo-aléatoire. La séquence de nombres produite est toujours la même, à moins d'introduire ce que l'on appelle une semence en choisissant un point de départ au hasard dans la séquence. Cette technique permet de se rapprocher de quelque chose d'imprévisible, car la séquence est très longue. Le générateur aléatoire de Python utilise comme semence la durée écoulée depuis la mise sous-tension du système.

L'autre technique consiste à utiliser quelque chose de vraiment imprévisible. Nous choisissons de surveiller la durée d'appui sur le bouton pour déterminer le chiffre sur lequel va tomber le dé. À première vue, cette action ne semble pas très aléatoire, car le joueur pourrait décider d'appuyer longtemps ou pas, afin d'influencer le tirage au sort. En réalité, le processeur peut mesurer le temps d'appui de manière extrêmement précise. Il va balayer les six valeurs possibles plusieurs milliers de fois par seconde. Aucun humain ne peut contrôler la durée d'appui sur le bouton avec une telle précision. Nous avons donc une bonne source de valeur aléatoire. Dans le projet, nous allons générer le tirage en comptant en boucle de 1 à 6 tant que le bouton est enfoncé.

Un problème technique que nous devons gérer est le problème du rebond des contacts du bouton-poussoir. Lorsque l'on appuie sur un contacteur, le contact ne se fait pas définitivement dès la première fois ; il y a des micro-interruptions, un peu comme une balle de ping-pong qui rebondit. Tous les interrupteurs mécaniques ont ce problème, mais certains en souffrent plus que d'autres. Certains projets ont d'autres actions à réaliser pendant l'attente de fin de rebond, ce qui réduit le problème, mais dans d'autres projets, il faut gérer le rebond. La solution est très simple : il suffit d'attendre quelques millisecondes après le premier contact avant de considérer que la valeur que nous voulons lire est stabilisée. C'est cette technique qui s'appelle l'antirebond d'entrée. Nous verrons comment programmer cela dans cet exemple.

Afficher le nombre

Une fois que nous avons tiré notre valeur au sort, nous devons l'afficher selon le motif d'un dé. Pour chaque valeur entre un et six, il faut allumer certaines diodes LED mais pas les autres. La solution triviale consiste à créer six blocs conditionnels basés sur l'instruction **`if`**. Dans chaque bloc,

nous décidons d'allumer ou d'éteindre chacune des diodes LED. L'extrait que montre le Listing 16.7 (non exécutable) montre comment écrire ce genre de code conditionnel avec la librairie GPIO Zéro.

Listing 16.7 : Extrait de code pour contrôler l'allumage des LED.

```
# La variable nombre contient la valeur a
afficher
if nombre == 3:   # Motif du trois
   led0.off()
   led1.on()        # Un coin
   led2.off()
   led3.on()        # Le milieu
   led4.off()
   led5.on()        # Un coin en diagonale
   led6.off()
if nombre == 4:   # Motif du quatre
   led0.on()
   # et ainsi de suite pour les autres nombres
```

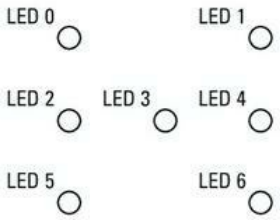
Cet extrait est facile à lire, mais qu'est-ce qu'il est ennuyeux et inutilement long. Rappelons qu'il faut mettre en place six blocs conditionnels comme celui du 3, un pour chaque valeur du dé. C'est typiquement le travail d'un débutant.

Cette façon d'écrire très peu optimisée n'est pas liée à l'utilisation de la librairie GPIO Zéro. Vous pouvez écrire du mauvais code avec n'importe quelle librairie et n'importe quel langage de programmation.

Vous pourriez vous demander si cela a tant d'importance, à partir du moment où le code fonctionne. En fait, si vous apprenez à écrire proprement, vous pourrez appliquer ces nouvelles techniques dans des situations dans lesquelles l'approche du débutant ne fonctionne plus du tout.

Voyons maintenant comment faire pour décider quelles diodes LED doivent être allumées et lesquelles ne doivent pas l'être en fonction de la valeur. Le tableau de la Figure 16.12 montre les motifs à afficher. Les deux choix essentiels concernent l'affichage des valeurs 3 et 2 qui utilisent des diagonales. Nous avons choisi d'utiliser les deux diagonales

opposées pour chacun des deux chiffres, mais nous n'avons pas de préférence quant au choix entre diagonale gauche et diagonale droite.



Valeur affichée	LED 6	LED 5	LED 4	LED 3	LED 2	LED 1	LED 0	Motif binaire	Motif hexa	Motif décimal
1	off	off	off	on	off	off	off	0001000	0x08	8
2	on	off	off	off	off	off	on	1000001	0x41	65
3	off	on	off	on	off	on	off	0101010	0x2A	42
4	on	on	off	off	off	on	on	1100011	0x63	99
5	on	on	off	on	off	on	on	1101011	0x6B	107
6	on	on	on	off	on	on	on	1110111	0x77	119

Figure 16.12 : Codage des motifs d'affichage du dé.

Observez bien dans le tableau la colonne montrant le motif binaire. En comptant les sept LED de zéro à six, ce motif permet de savoir quelle LED est allumée pour chaque valeur. Chaque chiffre du motif correspond à un bit, c'est-à-dire à une valeur élémentaire telle que stockée dans l'ordinateur. Cela permet de coder le motif d'affichage des diodes LED. Dans notre programme, nous allons utiliser ce motif en le convertissant dans sa valeur hexadécimale, c'est-à-dire en base 16. Chaque chiffre hexadécimal, de 0 à F, regroupe quatre bits. Si vous lisez la colonne du motif hexa, vous constatez qu'il y a d'abord le préfixe 0x, qui prévient le programme que la valeur est exprimée en base hexadécimale, puis il y a deux chiffres ou lettres. Pour la valeur de dé 2, la valeur hexa est égale à 41, c'est-à-dire à quatre fois 16 ou 64 plus 1, c'est-à-dire 65, ce que confirme la valeur décimale dans la dernière colonne.

La colonne qui indique le motif en valeur décimale correspond à la notation que nous avons apprise à l'école. Cette notation décimale est la moins intéressante pour répondre à notre besoin, parce qu'il est très difficile de définir un motif binaire à partir d'une valeur décimale, alors que c'est beaucoup plus simple à partir d'une valeur hexadécimale.

Rappelons que la valeur numérique, que ce soit en hexa ou en décimal, n'a aucune importance. Elle n'est là que pour coder le motif binaire de zéros et de un qui correspond à l'état éteint ou allumé de chaque diode LED. Souvent, les débutants demandent comment convertir une valeur en binaire, alors que c'est inutile parce que toutes les valeurs numériques sont stockées en binaire dans les ordinateurs. C'est au contraire lorsque vous avez besoin de communiquer avec un humain, en affichant un message ou une valeur, qu'il faut reconvertir depuis le binaire vers le décimal, ce que fait automatiquement la fonction **`print()`**. Pour notre exemple, il nous suffit de savoir exploiter ce motif binaire pour décider d'allumer ou d'éteindre les diodes LED. Autrement dit, il faut transformer la valeur en un véritable motif, et nous allons tenter de le faire de manière efficace.

Commençons en douceur en voyant comment traiter un seul bit du motif, par exemple le bit le moins significatif, c'est-à-dire celui de droite dans le motif binaire. Si ce bit vaut un, nous allumons la diode LED ; s'il vaut zéro, nous l'éteignons. Voici le genre d'instruction qui nous permettrait d'y parvenir :

```
io.output(numLED, motif & 1)
```

Vous aurez deviné que la variable qui porte le nom `numLED` contient le numéro de la diode LED que vous voulez contrôler. De même, la variable `motif` contient le motif de bits que vous voulez reproduire sur les diodes. Le symbole & correspond à l'opération ET logique entre bits. Cet opérateur travaille une position binaire à la fois. Par exemple, pour le bit le plus à droite, s'il y a un 1 dans les deux nombres, le résultat sera à 1. Dans les trois autres cas, le résultat sera à 0. Le fait d'appliquer le ET logique entre le motif et la valeur 1 a pour résultat que toutes les positions binaires du motif sont forcées à zéro, sauf le bit le plus à droite qui nous intéresse ici, qui est inchangé. C'est une manière d'isoler un bit dans un groupe de bits, représenté par la variable `motif`. Si le premier bit de droite vaut 0, l'opération renvoie 0. S'il vaut 1, l'opération renvoie 1. Cette valeur est directement celle qu'il faut indiquer dans la commande de contrôle de la diode LED. La valeur combinée au motif par le ET sert de masque, parce qu'elle masque tous les autres bits qui ne nous intéressent pas.

Le numéro de la diode LED doit changer à chaque fois que nous traitons un bit différent du motif. Nous allons à cet effet utiliser une liste qui contient les numéros des broches GPIO correspondant aux différentes diodes LED. Le premier élément de la liste correspond à la diode 0, c'est-à-dire la broche 23. Et ainsi de suite. Il nous reste à trouver une technique pour décaler le motif binaire dans la variable `motif` d'une position vers la droite afin que la même instruction de masquage avec le ET logique puisse traiter à chaque fois le bit le plus à droite. Il suffit d'utiliser l'opérateur de décalage vers la droite qui correspond à deux signes **>>**. Voyons la petite boucle qui permet de générer le motif :

```
numLED = [23,4,25,10,17,8,22]
for i in range(0,len(numLED));
  io.output(numLED[i], motif & 1)
  motif = motif >> 1
```

Nous n'avons que deux instructions dans notre boucle, mais cela suffit à traiter la totalité du motif ! C'est un code bien plus compact que le

mauvais exemple vu un peu plus haut.

Pour le plaisir, nous avons ajouté une fonction qui simule le fait que le dé roule sur la table. Elle consiste à afficher plusieurs motifs choisis au hasard et se succédant rapidement.

Nous sommes maintenant prêts à découvrir la totalité du code source, ce que propose le Listing 16.8.

Listing 16.8 : Code source complet du dé électronique.

```
#!/usr/bin/python3
# Electronic dice. par Mike Cook
import time, random
import RPi.GPIO as io

numLED     = [23,4,25,10,17,8,22]
aleaMotifs = [0,0x08,0x41,0x2A,0x63,0x6B,0x77]
inter  = 7

def main():
   print("DE ELECTRONIQUE - Ctrl C pour quitter")
   initGPIO()
   nombre = 1
   while 1:
      afficherRouler()
      afficherNombre(nombre)
      nombre = genererNombre()

def afficherRouler():          # Suite de motifs
pour faire rouler
   for i in range(0,20):
     afficherNombre(random.randint(1,6))
     time.sleep(0.1)

def afficherNombre(nombre):
    motif = aleaMotifs[nombre]
    for i in range(0, len(numLED)):
```

```
        io.output(numLED[i], motif & 1)
        motif = motif >> 1

def genererNombre():     # Attente d'appui
    lancer = random.randint(1,6)
    while io.input(inter) == 1:
        pass
    time.sleep(0.030)    # Antirebond
    while io.input(inter) == 0:
        lancer += 1
        if lancer >6:      # Reboucle
            lancer = 1
    return lancer
def initGPIO():
    io.setmode(io.BCM)
    io.setwarnings(False)
    for pin in range (0,len(numLED)):
        io.setup(numLED[pin],io.OUT)      # pin
passe en sortie
    io.setup(inter,io.IN,pull_up_down=io.PUD_UP)
# pin redevient en entree

# Logique principale
if __name__ == '__main__':
    main()
```

En début d'exécution de ce code, nous chargeons les fonctions des librairies, et définissons les variables globales et toutes les fonctions qui seront appelées depuis la fonction principale **`main()`** qui sert de plaque tournante à la totalité du programme. Dans la mesure du possible, arrangez-vous pour qu'elle reste assez courte. Dans l'exemple, elle affiche un message, prépare les broches GPIO et la valeur à afficher d'abord, puis entre dans une boucle infinie qui affiche l'animation du dé qui roule puis la valeur trouvée, en enchaînant avec la génération du prochain nombre dès appui sur l'interrupteur.

Les débutants font souvent l'erreur de vouloir rassembler toutes les instructions dans la fonction **`main()`**. Il est pourtant beaucoup plus simple de structurer le traitement en plusieurs fonctions dont vous

choisissez le nom avec soin. Adoptez une convention pour donner les noms à vos fonctions. N'utilisez pas de lettres accentuées ni de caractères spéciaux. En choisissant les noms avec soin, vous allez faciliter la lecture aux autres, et aussi à vous-même lorsque dans plusieurs mois vous aurez besoin de relire ce code source.

Une chose qui peut étonner certains lecteurs est le fait que la liste qui contient les motifs commence avec un 0, alors que cette valeur n'est pas présente dans le tableau de la Figure 16.12. En programmation, les comptages et les listes commencent généralement à zéro. Pour le motif, le premier qui nous intéresse correspond à la valeur un. C'est pourquoi nous avons ajouté une entrée vide, valant zéro, pour que la position du motif soit alignée avec la position dans la liste. Autrement dit, le motif pour la valeur 4 est bien à la quatrième position dans la liste. La suite du code permet de garantir que le zéro n'aura jamais besoin d'être affiché.

Toujours dans le Listing 16.8, vous voyez que la fonction **`afficherRouler()`** se résume à une boucle qui appelle 20 fois la fonction **`afficherNombre()`** en lui fournissant à chaque fois une valeur entre 1 et 6 choisie au hasard. Chaque valeur est affichée 1/10 seconde. La fonction **`afficherNombre()`** ne comporte rien de neuf par rapport aux exemples précédents du chapitre. En revanche, la fonction **`genererNombre()`** mérite quelques explications. Elle commence par choisir une valeur au hasard qu'elle stocke dans la variable `lancer` puis elle se place en attente de l'appui sur le bouton dans une boucle **`while`**. Dès que l'appui est détecté, nous imposons une attente de 30 ms pour ignorer tout rebond parasite et nous entrons dans une autre boucle **`while`** tant que le bouton n'a pas été relâché. Pendant que l'exécution reste dans cette deuxième boucle, nous augmentons la valeur de la variable `lancer` pour aller jusqu'à 6 puis repartir à 1. Nous n'avons plus qu'à envoyer la valeur sur laquelle nous sommes sortis de la boucle vers la fonction principale. L'appel à la fonction s'écrit ainsi :

```
nombre = genererNombre()
```

Nous sommes assurés de récupérer la valeur du dé dans la variable `nombre`.

Pour aller plus loin

N'hésitez pas à modifier le code source de nos exemples, et à l'enrichir. Vous pouvez par exemple sans trop de risques intervenir sur l'animation

du dé qui roule. Au lieu d'afficher un nombre au hasard, vous pouvez faire afficher des motifs prédéfinis. Pour vous aider à démarrer, nous proposons une autre manière d'écrire la fonction d'affichage du dé qui roule (Listing 16.9). Cet exemple vient remplacer la définition de la même fonction du Listing 16.8. Pensez à enregistrer le fichier sous un nouveau nom à chaque fois que vous faites une variante.

Listing 16.9 : Autre version de la fonction afficherRouler().

```
def afficherRouler():
   motifRouler =
[0x01,0x03,0x07,0x0F,0x1F,0x3F,0x7F,0,0]
   for rouler in range(0, len(motifRouler)):
      motif = motifRouler[rouler]
      for i in range(0,len(numLED)):
         io.output(numLED[i], motif & 1)
         motif = motif >> 1
      time.sleep(0.2)
```

Nous avons défini ici une nouvelle série de motifs dans la variable `motifRouler`. Vous pouvez indiquer d'autres valeurs dans la liste pour changer l'animation. Les instructions garantissent que tous les motifs seront affichés. Sauriez-vous trouver les valeurs indiquées dans la liste pour qu'une seule diode LED s'allume à la fois, comme si elle faisait le tour du dé ? Notez que vous pouvez également influer sur la vitesse d'affichage, par exemple en ralentissant le changement vers la fin de l'animation. Vous pouvez bien sûr utiliser des valeurs binaires pour les motifs, comme ceci :

```
motifRouler =
[0b1,0b11,0b111,0b1111,0b11111,0b111111,0b1111111,
```

Il faut reconnaître qu'il faut un peu d'habitude pour parvenir à lire des motifs binaires sans s'y perdre. Si ce projet vous plaît, vous pouvez le placer dans une boîte translucide et le relier à votre carte RasPi avec un câble en nappe.

Simulateur de passage piéton

Le projet de dé électronique nous a permis de voir comment gérer des motifs binaires. Découvrons maintenant un projet qui s'intéresse à la temporisation et aux séquences. Il s'agit d'un simulateur de passages piétons tel qu'ils sont connus au Royaume-Uni. Il s'agit d'un feu de circulation tricolore et d'un feu de passage piéton contrôlé par un bouton. Les détails de temporisation sont différents en France, mais le principe reste applicable. Découvrons donc la séquence que nous devons simuler.

Le feu de circulation tricolore est constitué d'une diode rouge, d'une diode orange et d'une diode verte. Le feu du passage piéton n'a que deux couleurs : vert pour passer et rouge pour patienter. L'état normal du système est le feu vert pour la circulation, tous les autres feux éteints. Le bouton permet à un piéton de provoquer le passage au rouge. Un capteur de trafic est prévu sous la chaussée. Ce capteur permet d'ajouter un délai avant prise en compte de la demande du piéton pour traverser. Si aucun trafic n'est détecté, le feu passe immédiatement au rouge. S'il y a du trafic, nous ajoutons un délai pour éviter que les piétons arrêtent la circulation trop fréquemment. La lumière rouge pour les piétons s'allume dès que l'on appuie sur le bouton. (En France, elle reste allumée en permanence.)

Au début de la séquence pour les piétons, le feu de circulation passe à l'orange puis au rouge. Le feu vert du piéton s'allume et un bruiteur émet un signal à destination des personnes malvoyantes. En fin de séquence, les piétons peuvent terminer leur traversée, mais pas commencer à traverser. Le feu piéton vert et le feu orange de circulation clignotent alors et le bruiteur s'éteint. Puis le feu rouge piéton s'allume pendant que le feu de circulation reste orange, avant de passer enfin au vert. À ce moment, les deux feux pour piétons s'éteignent.

Cette séquence est assez simple, mais il ne faut pas oublier de tester le capteur de trafic au cours de la séquence, afin de détecter l'arrivée de véhicules, et d'empêcher qu'un second appui sur le bouton fasse rester dans l'état passage piéton. Nous ne pouvons pas nous contenter d'ajouter un délai avec **`time.sleep`**. Nous devons mettre en place une machine à états, c'est-à-dire un mécanisme qui permet de traiter plusieurs processus comme s'ils s'exécutaient en parallèle.

Le concept d'automate fini est fondamental ; vous vous en servirez de nombreuses fois dans vos projets.

Un automate fini permet de gérer des processus qui ne réclament pas toute l'attention du processeur en permanence. Le programme de clignotement de diodes LED passe presque la totalité de son temps d'exécution dans les fonctions d'attente, attendant que le temps passe. Dans un automate fini,

on cherche à tirer profit de ce temps perdu en réalisant d'autres tâches. Mais il faut savoir à quel moment les tâches dont vous vous détournez vont à nouveau avoir besoin de votre attention. Il faut pour cela définir une variable dans laquelle est indiqué le temps disponible avant la prochaine exécution de chaque tâche. Nous utilisons à cet effet la méthode nommée **`time.time()`** de la librairie de même nom. Elle renvoie une valeur numérique avec partie décimale qui correspond au nombre de secondes écoulées depuis l'allumage du système. La valeur n'a aucune importance ; nous allons nous en servir de façon relative en ajoutant et soustrayant une durée.

Voyons cela avec un exemple simple : nous allons faire clignoter deux diodes LED indépendamment l'une de l'autre.

Commencez par effectuer le câblage très simple suivant. Vous utilisez les deux broches 23 et 24, comme le montre la Figure 16.13.

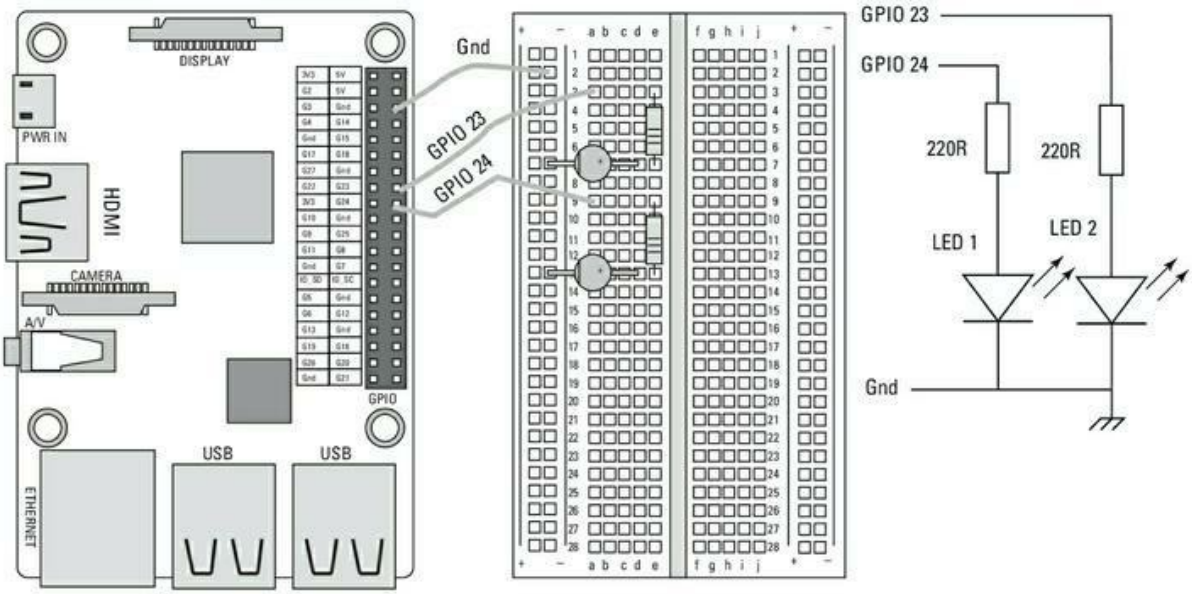


Figure 16.13 : Câblage de deux diodes LED pour un essai de clignotement indépendant.

Voyons comment faire clignoter les deux diodes de façon totalement dépendante. Chaque tâche est gérée par une variable d'état qui peut avoir la valeur 0 ou 1. Cette variable permet de savoir si la diode est actuellement allumée ou éteinte. Vous savez ainsi quoi faire lors de la prochaine exécution de la tâche. Le délai entre deux changements d'état d'une diode est déterminé par une variable qui est le fruit d'un calcul consistant à ajouter l'heure actuelle au délai désiré entre deux changements d'état. Cette valeur peut être différente pour chaque état. Dans le Listing 16.10, le délai est le même pour les deux états de la LED1, mais pas pour les deux états de la LED2.

Listing 16.10 : Deux clignotements indépendants.

```
#!/usr/bin/python3
# Deux clignotements - par Mike Cook
import time
import RPi.GPIO as io

broLed1 = 23
broLed2 = 24
vitCliLed1 = 0.5    # Vitesse de cligno
tpsAlluLed2 = 0.5
tpsNoirLed2 = 0.51
etatLed1 = 0
etatLed2 = 0
tpsCligno1 = time.time()
tpsCligno2 = time.time()

def main():
   print("Deux clignos - Ctrl C pour quitter")
   initGPIO()
   while 1:
      if time.time() > tpsCligno1:
         clignoter1()
      if time.time() > tpsCligno2:
         clignoter2()

def clignoter1():
   global tpsCligno1, etatLed1
   if etatLed1 == 0:
      io.output(broLed1,1)
      etatLed1 = 1
    else:
      io.output(broLed1,0)
      etatLed1 = 0
   tpsCligno1 = time.time() + vitCliLed1
```

```
def clignoter2():
   global tpsCligno2, etatLed2
   if etatLed2 == 0:
      io.output(broLed2,1)
      etatLed2 = 1
      tpsCligno2 = time.time() + tpsAlluLed2

    else:
      io.output(broLed2,0)
      etatLed2 = 0
      tpsCligno2 = time.time() + tpsNoirLed2

def initGPIO():
   io.setmode(io.BCM)
   io.setwarnings(False)
   io.setup(broLed1,io.OUT)
   io.setup(broLed2,io.OUT)

# Main program logic:
if __name__ == '__main__':
   main()
```

Ce programme passe l'essentiel de son temps dans la boucle de répétition **`while`** de la fonction principale **`main()`**. Il cherche à savoir si le moment est venu d'appeler l'une ou l'autre des deux fonctions publiques **`clignoter1`** et **`clignoter2`**. Note : vous pouvez ajouter d'autres fonctions ici si nécessaire. Dès qu'une fonction est lancée, elle fait évoluer l'état de la tâche, c'est-à-dire qu'elle allume ou éteint la diode LED. Nous inversons la valeur de la variable `etatLedx` puis réglons l'heure de la prochaine exécution de la tâche. Pour la LED1, nous faisons cette reprogrammation à la fin de la fonction, mais pour la LED2, les deux durées étant différentes, nous faisons ce réglage à la fin de chaque branche conditionnelle.

Lorsque vous exécutez le programme, vous voyez au départ les deux diodes clignoter ensemble, mais elles finissent par se désynchroniser, pour au bout d'un certain temps se resynchroniser. C'est le fruit du choix d'une longueur légèrement supérieure pour l'état à 1 de la LED2 (0.51 au lieu de 0.50).

Essayez de modifier les valeurs des pauses pour en voir les effets. Sauriez-vous ajouter deux autres diodes LED à ce montage ?

Montage du circuit de passage piéton

Nous disposons maintenant d'assez d'informations techniques pour bien maîtriser le projet. Nous pouvons donc passer au montage du circuit. Inspirez-vous du schéma de principe de la Figure 16.14.

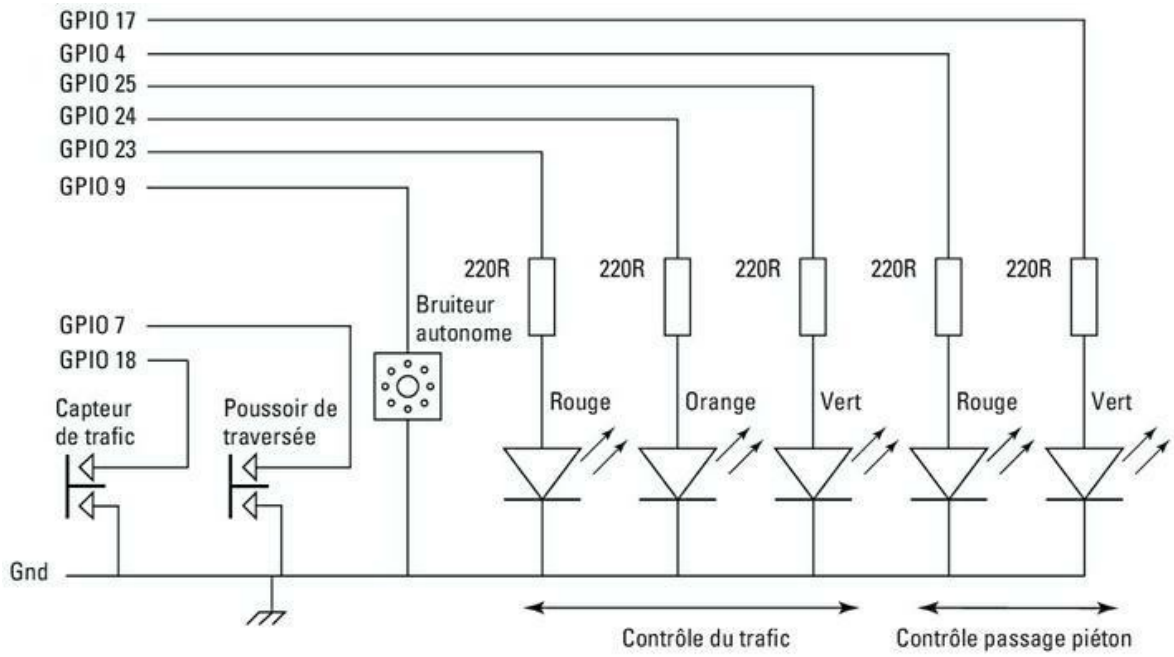


Figure 16.14 : Schéma de principe du passage piéton.

Ce montage ne s'éloigne pas beaucoup des précédents. Notez qu'il y a deux boutons interrupteurs, un pour le capteur de trafic et un pour la demande de traversée. Les cinq diodes LED sont munies de leur résistance. Si possible, vous choisirez deux diodes rouges, deux vertes et une orange. Le composant nouveau est le bruiteur (buzzer). Notez qu'il existe deux modèles de buzzers piézo-électriques. Les deux modèles sont souvent confondus, même par les revendeurs. Le modèle qu'il nous faut est celui qui produit un son dès que vous appliquez une tension continue. Nous appelons cela un bruiteur autonome. L'autre modèle doit être alimenté avec des impulsions variables pour émettre un son. Dans tous les cas, ne confondez pas ce genre de buzzer avec les buzzers électromécaniques qui consomment bien trop de courant pour notre fragile RasPi. Il vous faut un buzzer fonctionnant en 3 V et consommant moins de 10 mA. Une solution très simple pour vérifier que votre modèle est autonome, consiste à brancher une de ses entrées à une broche 3,3 V

puis à faire brièvement contact entre l'autre entrée et une broche de masse. Si vous n'entendez qu'un clic, c'est un buzzer haut-parleur, donc passif. Si vous entendez un vrai son, c'est un buzzer autonome. (Pour tout dire, nous avons remplacé le buzzer de notre montage par une diode LED avec sa résistance, car nous en avions plein les oreilles de faire des essais.)

La Figure 16.15 montre un exemple de plan de montage découlant de notre schéma de principe. Nous avons tenté d'installer les trois feux de circulation comme un feu tricolore, et de même pour les deux feux du passage piéton, comme dans la vraie vie.

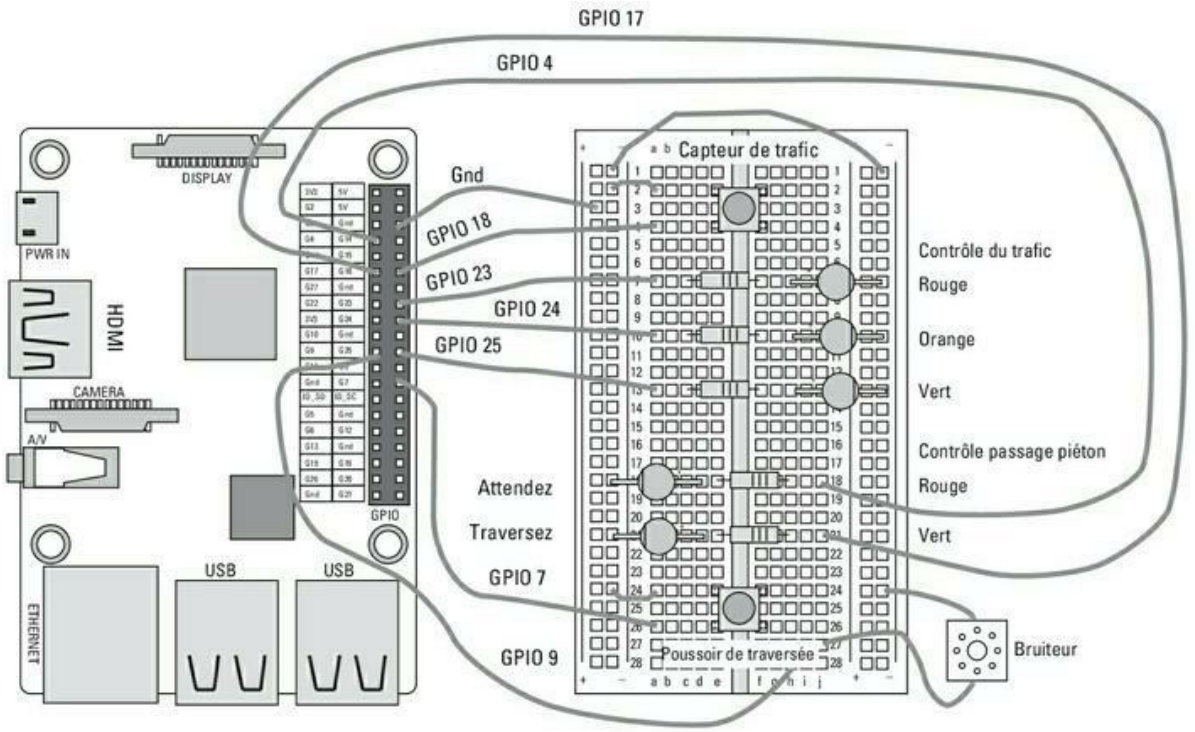


Figure 16.15 : Plan de montage du passage piéton.

Le logiciel du passage piéton

Découvrons maintenant comment tirer profit du concept d'automate fini dans notre simulateur de passage piéton. Nous devons prévoir une tâche pour surveiller le détecteur de trafic. Si nous détectons des véhicules, nous devons noter l'heure à laquelle nous avons effectué cette détection. Nous devons prévoir une autre tâche pour surveiller le bouton-poussoir de demande de traversée. Si le bouton est enfoncé mais que nous ne sommes pas dans la séquence de traversée, nous pouvons commencer la séquence. Enfin, il nous faut une autre tâche pour animer le bruiteur uniquement lorsque nous sommes dans la séquence de traversée des piétons. (Vous remarquez que nous avons réduit les durées dans la séquence de traversée,

cela pour simplifier les tests. Dans un vrai passage piéton, les durées seraient plus longues).

Découvrons maintenant le code source complet de ce projet, dans le Listing 16.11.

Listing 16.11 : Simulateur de passage piéton.

```
#!/usr/bin/python3
# Pedestrian crossing Par Mike Cook
import time, random
import RPi.GPIO as io

broLED = [23,24,25,4,17,9]
broAppel = 7
broCapTrafic = 18
hrSeqSuivante = time.time()

def main():
   global hrDerTrafic, etat, sonore
   print("Emulateur de passage pietons - Ctrl C
pour quitter")
   initGPIO()
   io.output(fVert,1) # Feu vert pour commencer
   etat = 0
   hrSonore = time.time()
   sonore = False
   hrDerTrafic = time.time()
   while 1:
      testerTrafic()
      if testerAppel() and etat == 0 :
         io.output(piStop,1)  # Allumer feu
attendre
         if time.time() - hrDerTrafic > 10.0: #
Traverser
            etat = 1
           else:
```

```
            time.sleep(10.0)   # Laisser passer
qqs vehicules
            etat = 1
      gererTraversee()
     if sonore and time.time() > hrSonore:
         hrSonore = time.time() +0.3

io.output(bruiteur,not(io.input(bruiteur)))

def testerTrafic():
   global hrDerTrafic
   if io.input(broCapTrafic) == 0:
      hrDerTrafic = time.time()

def testerAppel():
   request = False
   if io.input(broAppel) == 0:
      request = True
   return request

def gererTraversee():
   global hrSeqSuivante,nbrFlash,etat,sonore
   if etat == 0:
      hrSeqSuivante = time.time() + 2.0
      return
   if time.time() > hrSeqSuivante :
      if etat == 1:            # Feu orange
         # print("Etat actuel=",etat)
         io.output(fVert,0)
         io.output(fOrange,1)
         etat = 2
         hrSeqSuivante = time.time() + 2.0 #
Temps fOrange
      elif etat == 2:          # Feu rouge
         # print("Etat actuel=",etat)
         io.output(fOrange,0)
```

```
            io.output(fRouge,1)
            etat = 3
            hrSeqSuivante = time.time() + 2.0 #
Temps fRouge
         elif etat == 3:          # Feu pietons
            # print("Etat actuel=",etat)
            io.output(piStop,0)
            io.output(piTrav,1)
            sonore = True
            etat = 4
            hrSeqSuivante = time.time() + 5.0 #
Temps trav.
         elif etat == 4:         # Orange pour fin
pietons
            # print("Etat actuel=",etat)
            io.output(fOrange,1)
            io.output(fRouge,0)
            sonore = False
            io.output(bruiteur,0)     # Stop
bruiteur
            etat = 5
            hrSeqSuivante = time.time() + 1.0
         elif etat == 5:            # Cligno orange
et pietons
          # print("Etat actuel=",etat)

io.output(fOrange,not(io.input(fOrange)))
            io.output(piTrav,not(io.input(piTrav)))
            hrSeqSuivante = time.time() + 0.2 #
Vitesse cligno
            nbrFlash +=1
            if nbrFlash > 20 : # Temps trav. 20 *
vit. cligno
               nbrFlash = 0
               etat = 6
         elif etat == 6: # fOrange
```

```
            io.output(piTrav,0)
            io.output(piStop,1)
            io.output(fOrange,1)
            etat = 7
            hrSeqSuivante = time.time() + 2.0 #
Temps fOrange
         elif etat == 7:
            #print("Etat actuel=",etat)
            io.output(fOrange,0)
            io.output(fVert,1)
            io.output(piStop,0)
            etat = 0
            hrSeqSuivante = time.time() + 1.0

def initGPIO():
   global fRouge, fOrange, fVert, piStop, piTrav,
nbrFlash, bruiteur
   io.setmode(io.BCM)
   io.setwarnings(False)
   for pin in range (0,len(broLED)):
      io.setup(broLED[pin],io.OUT)
# Sortie
      io.output(broLED[pin],0)
# A zero

io.setup(broAppel,io.IN,pull_up_down=io.PUD_UP) #
Entree

io.setup(broCapTrafic,io.IN,pull_up_down=io.PUD_UP

   fRouge = broLED[0]
   fOrange = broLED[1]
   fVert = broLED[2]
   piStop = broLED[3]
   piTrav = broLED[4]
   bruiteur = broLED[5]
```

```
    nbrFlash = 0

# Main program logic:
if __name__ == '__main__':
    main()
```

Une première lecture du programme vous permet de constater que les broches GPIO possèdent chacune un nom différent, ce qui simplifie la lecture. (Nous aurions pu utiliser un nombre magique, en fait un indice pour accéder à une liste des broches GPIO.) Dans la boucle infinie **`while`** de la fonction principale **`main()`**, nous commençons par détecter le trafic puis le bouton-poussoir. La fonction **`testerAppel()`** renvoie la valeur True si le bouton a été enfoncé. Nous appelons cette fonction directement dans l'instruction conditionnelle **`if`**, sans perdre de temps à passer par une variable intermédiaire. L'instruction **`if`** vérifie également si une traversée est en cours grâce à la variable `etat`. Cette variable contient zéro s'il n'y a pas de traversée de piétons en cours. Dans le cas contraire, nous imposons un délai si nous avons détecté de la circulation dans les dix dernières secondes, puis nous forçons la variable `etat` à un.

De retour dans la fonction principale, nous appelons la fonction **`gererTraversee()`** qui rend immédiatement le contrôle par **`return`** si la variable `etat` vaut zéro. Dans le cas contraire, elle teste s'il faut changer la valeur de `etat` si le moment est venu le faire. Elle rend compte si ce n'est pas le cas. Nous testons enfin la variable `bleep` qui détermine l'activité du bruiteur. Cette variable est armée dans l'état 3 de la fonction **`gererTraversee()`** puis remise à zéro dans l'état 4. Cette fonction de gestion de traversée se charge pour l'essentiel d'allumer et d'éteindre les différentes diodes en fonction de la position dans la séquence. La seule exception correspond à l'état 5, dans lequel la lumière de traversée des piétons et le feu orange doivent clignoter. Nous y parvenons en basculant entre les deux lumières. Le basculement signifie qu'une des diodes est allumée pendant que l'autre est éteinte. Au lieu d'utiliser une variable pour se souvenir dans quel état était la diode lors du dernier tour, vous pouvez utiliser une petite astuce qui consiste à lire l'état de la broche de sortie GPIO. Cela vous donne directement la dernière valeur que vous lui aviez imposée, et vous savez qu'il suffit d'inverser cet état. La ligne suivante inverse par exemple l'état de la diode LED orange :

```
io.output(fOrange, not(io.input(fOrange)))
```

Pour passer de l’état 5 à l’état 6, il faut avoir fait clignoter 10 fois ces deux diodes. Dans l’état 7, vous remettez au vert le feu de circulation puis vous ramenez l’automate fini dans l’état 0, ce qui confirme que nous ne sommes plus dans la séquence de traversée.

Le code source comporte un certain nombre d’instructions d’affichage **`print()`** qui sont neutralisées par mise en commentaire avec un signe dièse initial. Vous pouvez les décommenter pour voir les différentes étapes s’enchaîner sans avoir besoin des diodes LED.

Vous pouvez sortir dans la rue pour chronométrer les durées en vigueur sur un vrai passage piéton, puis réutiliser ces durées dans votre projet. Vous pouvez aussi enrichir le projet pour le rendre plus intelligent, par exemple en comptant le nombre de voitures qui passent pendant un certain laps de temps, par exemple 20 secondes, afin de faire varier le délai avant le début de séquence de traversée en fonction du trafic.

Chapitre 17

Utiliser des diodes LEDRGB

DANS CE CHAPITRE :

- **Les diodes LED tricolores (RGB)**
- **Écrire une classe pilote APA102**
- **Le jeu de tir Rainbow Invaders**
- **Le jeu de jonglage Keepy Happy**
- **Rubans et matrices de LED**

Des LED de toutes les couleurs

Nous avons vu dans le chapitre précédent comment contrôler l'allumage de plusieurs diodes LED avec des motifs et des séquences. Nous n'avons utilisé que des diodes LED monochromes, chacune ne brillant que d'une couleur. Découvrons maintenant les diodes LED multicolores. Dans le projet de passage piéton du chapitre précédent, nous avons utilisé des diodes de couleurs différentes. Il existe aussi des diodes qui réunissent dans le même boîtier une LED de chacune des trois couleurs primaires. On les appelle les LED RGB parce qu'elles réunissent une diode LED Rouge, une diode LED verte (le G de *green* en anglais) et une diode LED Bleue.

Il existe trois types de brochage des sorties d'une diode RGB : à anode commune aux trois diodes, à cathode commune et les diodes plus rares dont chacune des trois diodes possède ses deux broches indépendantes. Ce dernier type se présente sous forme d'un petit rectangle à six broches destiné à être soudé en surface (SMC). C'est ce type qui est utilisé dans la carte d'extension Pimoroni Sense HAT présentée à la fin du Chapitre 15. La Figure 17.1 montre le brochage et les branchements internes des trois types de LED RGB.

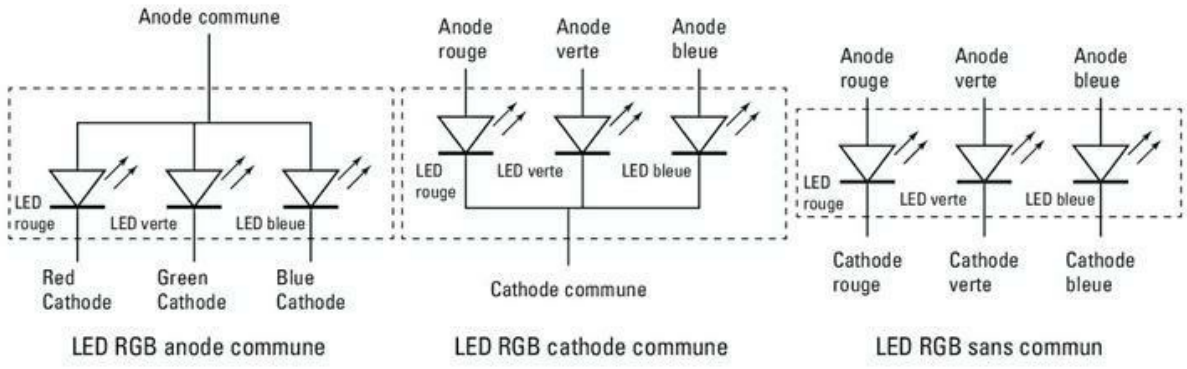


Figure 17.1 : Les trois types de brochage d'une LED RGB.

Le rectangle en pointillé rappelle que les trois diodes sont réunies dans le même boîtier physique. Vous constatez que les deux modèles à gauche et au centre n'ont que quatre broches de sortie, de par l'anode ou la cathode commune. Avec ce mode de câblage, il reste possible de contrôler indépendamment chacune des trois diodes par programme. La Figure 17.2 montre comment utiliser chacun des deux types de LED RGB.

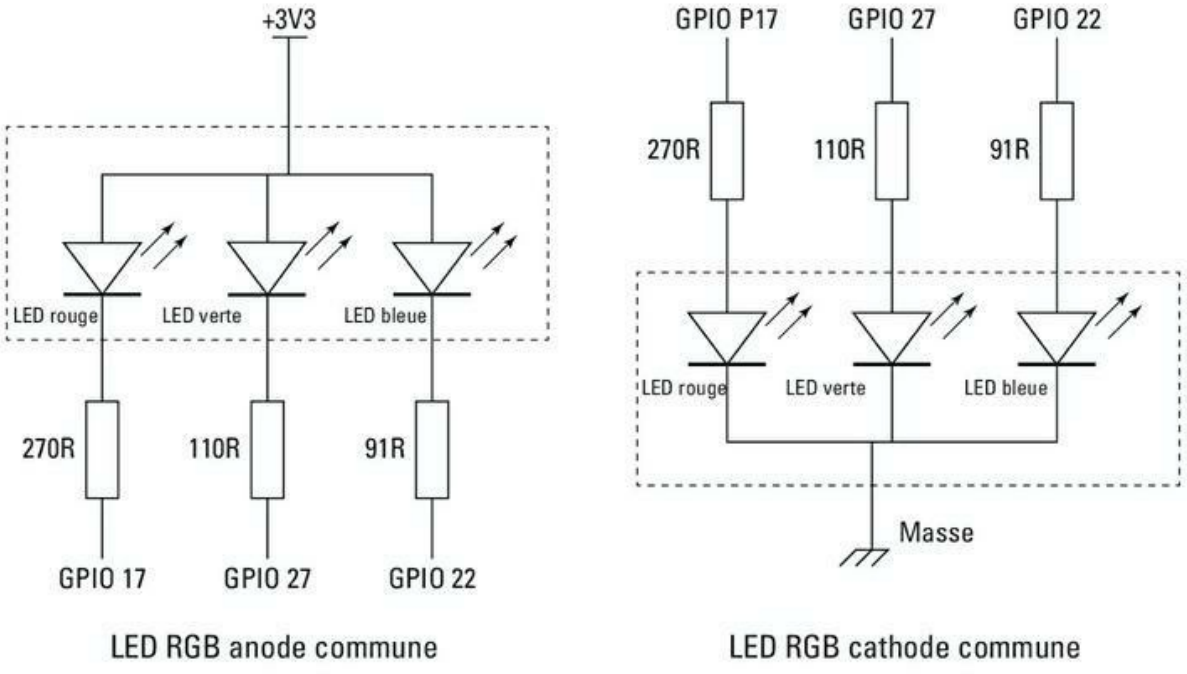


Figure 17.2 : Câblage d'une LED RGB à anode et à cathode commune.

Pour utiliser une diode à anode commune, il faut que les broches du connecteur GPIO drainent le courant vers la masse (*current sink*). Inversement, pour une diode à cathode commune, les broches GPIO doivent fonctionner en source de courant (*current source*), c'est-à-dire présenter la tension d'alimentation positive pour allumer la diode. (Vous pouvez revoir le Chapitre 16 pour un rappel des valeurs binaires.) Dans les deux cas, un débutant peut se demander, à la lecture de la Figure 17.2,

pourquoi il faudrait trois résistances. Ne serait-il pas possible de n'en utiliser qu'une seule sur la partie commune ? La réponse est claire : il faut absolument trois résistances, et voici pourquoi.

Si vous n'utilisez qu'une résistance, la chute de tension que va provoquer cette résistance sera différente selon le courant que va consommer la diode que vous cherchez à allumer. Autrement dit, la luminosité des LED ne sera pas identique. Si par exemple la LED rouge est allumée, il n'y a plus assez de tension pour allumer également les deux autres LED. En effet, selon la couleur de la LED, la chute de tension aux bornes n'est pas la même, et c'est pourquoi non seulement nous prévoyons trois résistances, mais leurs valeurs sont adaptées à la consommation de chaque LED.

N. d. T. : Voici l'affectation de chacune des quatre broches ou pattes d'une LED RGB :

- la broche la plus longue est celle de l'anode ou de la cathode commune ;
- la broche seule d'un côté de la plus longue est celle du rouge ;
- de l'autre côté, en partant de la plus longue, nous trouvons le vert puis le bleu (qui est la plus courte).

Mélangeons les couleurs

Puisque les problèmes de brochage sont maintenant réglés, nous pouvons passer à la partie amusante : comment mélanger les couleurs. Vous pouvez bien sûr allumer une diode LED à la fois, vous obtiendrez du rouge, du vert ou du bleu. Si vous en allumez deux, vous commencez à faire de la synthèse additive. Les diodes LED correspondent aux trois couleurs primaires rouge, vert et bleu. En les combinant deux à deux, vous obtenez des couleurs secondaires. Si vous allumez les diodes rouge et verte, vous obtenez du jaune. Le vert et le bleu donnent du cyan, et le rouge additionné au bleu donne du magenta. Si vous allumez les trois diodes, magie, vous obtenez du blanc, une sorte de blanc du moins.

En théorie, le fait d'allumer les trois diodes doit donner du blanc, mais la teinte exacte dépend de la luminosité relative des trois diodes. Pour obtenir du vrai blanc, il faut que les trois diodes soient exactement

équilibrées. Dans tous les autres cas, le blanc sera teinté d'après la diode prédominante. L'équilibre n'est pas facile à trouver parce que chaque diode provoque une chute de tension spécifique. Il faut choisir avec soin la valeur des résistances pour obtenir la même quantité de courant dans chaque diode. Et cela ne suffira même pas, car en fonction de leur couleur, les diodes ont un rendu différent, ce qui complique encore les choses.

Vous avez bien sûr appris à mélanger des couleurs avec de la peinture, mais il s'agit là de synthèse soustractive. Avec de la gouache, lorsque vous ajoutez du rouge et du vert à du bleu, vous finissez par obtenir un marron sombre. C'est selon cette technique que fonctionne l'impression en couleur. Les trois couleurs primaires de la synthèse soustractive sont les trois couleurs secondaires de la synthèse additive, c'est-à-dire le cyan, le magenta et le jaune. Les couleurs primaires en synthèse additive sont le rouge, le vert et le bleu.

Lumière diffuse ou non

Les diodes LED RGB sont généralement vendues en boîtier dépoli, ce qui permet de mieux fusionner les trois sources de couleur pour n'obtenir qu'une nuance. Le fait de s'éloigner suffisamment des trois LED RGB permet de ne voir que la couleur résultante, mais pour que cette fusion se fasse lorsque vous êtes proche de la diode, il faut soit que le boîtier soit dépoli, soit y ajouter par exemple de petits morceaux de papier calque.

Si inversement vous avez besoin que les trois couleurs restent bien distinctes, il faut soit vous procurer une diode RGB non dépolie, soit, et c'est encore plus efficace, revenir à trois diodes monochromes indépendantes. Si vous augmentez le taux de diffusion d'une diode en ajoutant par exemple du calque, tenez compte du dégagement de chaleur éventuel des diodes pour que l'ensemble ne provoque pas un incendie. En général, il suffit d'écarter le calque de quelques millimètres.

Vous pouvez convertir une diode LED claire en diode dépolie en appliquant un peu de papier de verre sur le boîtier. Un bloc à poncer sera encore plus efficace, car il pourra mieux épouser les courbes. Les boîtiers des diodes LED sont faits en résine ; les solvants tels que l'acétone n'ont pas d'effet sur leur surface.

Une vraie palette de couleurs

Pour obtenir plus que les sept teintes que sont les trois couleurs primaires, les trois couleurs secondaires et le blanc, il faut varier la luminosité

relative de chacune des trois diodes. Vous obtenez ainsi toute une palette de nuances. Mais comment faire pour contrôler la luminosité d'une diode ? La technique consiste à allumer et éteindre la diode à une fréquence telle que l'œil humain ne peut pas détecter l'astuce, c'est-à-dire au moins 30 fois par seconde. La proportion entre le temps allumé et le temps éteint va directement déterminer la luminosité apparente. Si la diode est allumée et éteinte pendant les mêmes durées, elle offrira une luminosité moyenne. Cette technique de commutation rapide porte le nom de modulation de largeur d'impulsion ; on parle plus généralement de PWM (*Pulse Width Modulation*). La Figure 17.3 montre trois cas : luminosité maximale, moyenne et minimale.

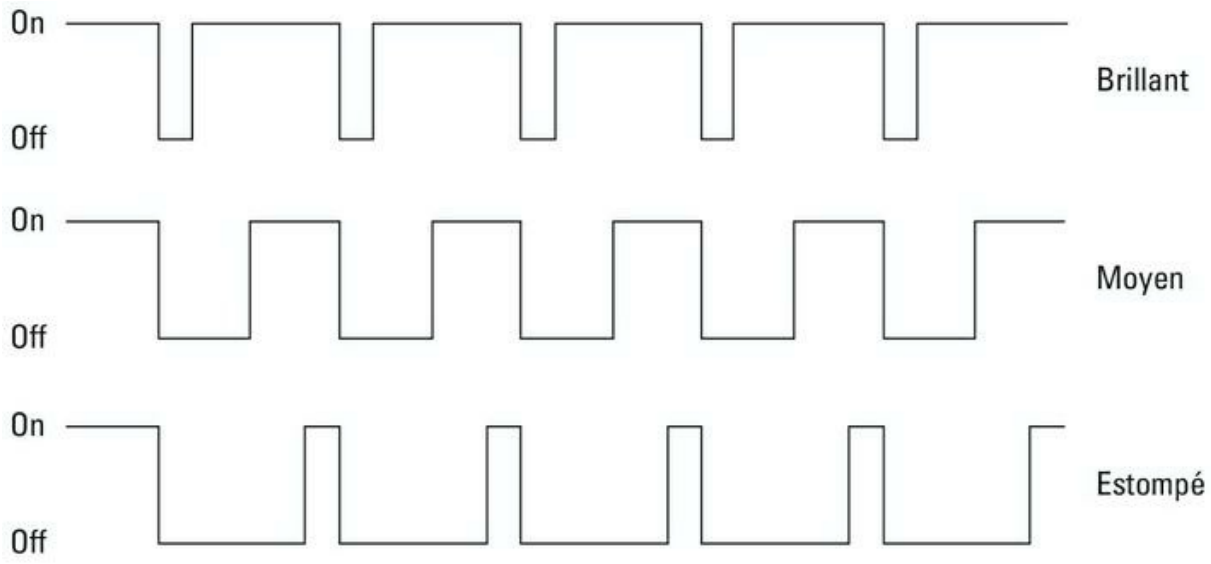


Figure 17.3 : Principe de la modulation de largeur d'impulsion PWM.

Vous pouvez voir que la fréquence des trois signaux PWM reste la même. Vous obtenez une grande luminosité en passant plus de temps dans l'état allumé que dans l'état éteint. Pour une lumière faible, c'est le contraire. La proportion entre état allumé et état éteint correspond au *rapport cyclique* (*duty cycle*). La fréquence exacte du signal n'a pas d'importance à partir du moment où elle est supérieure à ce que l'œil est capable de distinguer comme clignotement.

La librairie RPi. GPIO permet d'utiliser les broches GPIO en sortie avec un signal par impulsion PWM. Les fonctions correspondantes permettent de choisir la fréquence et le cycle de charge. Vous pouvez réaliser un montage de diode RGB comme visible dans la Figure 17.2, à anode ou cathode commune, puis tester votre palette de couleurs au moyen du code donné dans le Listing 17.1.

Listing 17.1 : Test de couleur RGB.

```
#!/usr/bin/python3
import time
import RPi.GPIO as io

io.setmode(io.BCM)
io.setup(17,io.OUT)       # Configuration en
sortie
io.setup(27,io.OUT)
io.setup(22,io.OUT)

ledR = io.PWM(17,60)      # Frequence PWM = 60Hz
ledG = io.PWM(27,60)
ledB = io.PWM(22,60)

ledR.start(0)             # Lancement PWM
ledG.start(0)
ledB.start(0)

print("Balayage de LED RGB avec RPi.GPIO - Par
Mike Cook")
print("Ctrl C pour quitter")
try:
   while(1):
       print("Debut de cycle")
       time.sleep(2.0)
       for stepR in range(0,100,5):             #
20 tours
          for stepG in range(0,100,5):          #
400 tours
             for stepB in range(0,100,5):      #
8000 tours
                ledR.ChangeDutyCycle(stepR)
                ledG.ChangeDutyCycle(stepG)
                ledB.ChangeDutyCycle(stepB)
                time.sleep(0.1)                 #
8000 fois
```

```
                    ledR.ChangeDutyCycle(0)
                    ledG.ChangeDutyCycle(0)
                    ledB.ChangeDutyCycle(0)

except KeyboardInterrupt:
   pass

ledR.stop(0)      # Arret PWM
ledG.stop(0)
ledB.stop(0)
io.cleanup()      # Restaure les etats GPIO
initiaux
```

En étudiant ce code source, nous voyons qu'il commence par configurer trois broches GPIO en sortie puis qu'il les prépare à produire un signal pulsé PWM. La valeur 60 qui sert de second paramètre est une fréquence, celle à laquelle doit être émis le signal PWM. Le rapport cyclique va de 0 à 100, donc d'éteint à allumé en permanence. La partie principale du programme est un jeu de trois boucles conditionnelles imbriquées qui permet de visiter toutes les combinaisons possibles des trois couleurs primaires rouge, vert et bleu, en augmentant le rapport cyclique de cinq à chaque étape. Un cycle complet dure ainsi 800 secondes, soit un peu plus de 13 minutes, et permet d'afficher 8000 nuances différentes. Lorsque vous testez le code, vous n'avez peut-être pas l'impression de voir tant de nuances, mais elles sont pourtant générées. Cette impression vient du fait que l'œil humain ne parvient pas très bien à distinguer des nuances proches, car il est sensible à des différences de couleurs beaucoup plus franches.

Des LED intelligentes

Le problème des diodes LED individuelles est que leur contrôle occupe beaucoup de temps de traitement du processeur du RasPi, et un grand nombre de broches de sortie GPIO, puisque vous en occupez trois par diode. Ce problème a été résolu de façon élégante voici quelques années lorsqu'ont été lancées les diodes LED équipées de leur propre générateur d'impulsions PWM. Il suffit d'envoyer un signal à ce genre de LED pour lui dire quelle couleur elle doit afficher avec quelle luminosité. La LED se charge du maintien de cet état jusqu'à ce que vous lui transmettiez un autre ordre. Mieux encore, ce genre de diode peut être chaînée sous forme de guirlande. Lorsque la première diode LED a reçu la commande qui la

concerne, elle transmet toutes les instructions suivantes qui lui arrivent à la diode LED suivante dans la chaîne.

Il existe deux technologies principales pour ce genre de diode LED : WC2812b et APA102. La société Adafruit vend le premier modèle sous le nom NeoPixels et le second modèle sous le nom DotStar. De façon générique, il s'agit de diodes LED adressables. Notez au passage que le modèle SK9822 est identique au APA102C.

L'aspect physique de ces diodes LED est particulier : elles se présentent sous forme de petits carrés de 5 mm de côté (Figure 17.4). Au centre de chaque carré, vous pouvez deviner le petit circuit noir qui combine le générateur d'impulsions et son circuit mémoire (qui mémorise la couleur demandée). Les zones translucides sont celles qui émettent la lumière.

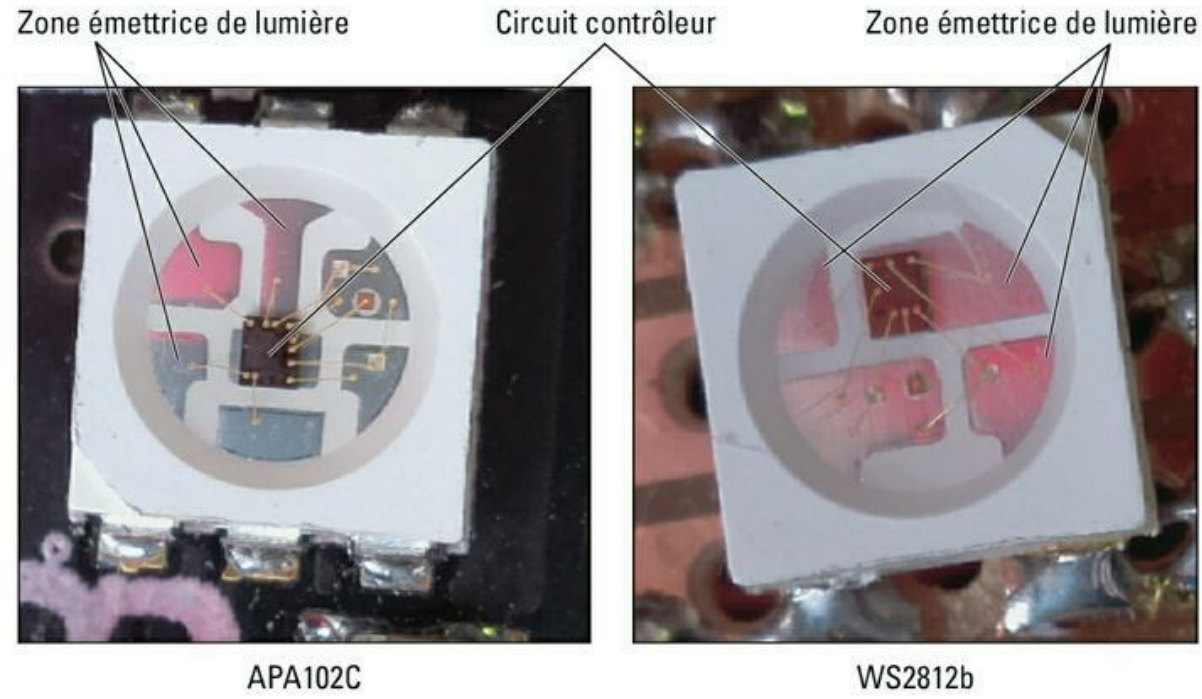


Figure 17.4 : LED adressables DotStar APA102 et NeoPixels WC2812b.

Le modèle NeoPixels est également disponible sous forme de boîtier classique avec des pattes, c'est-à-dire dans un format plus accessible que celui monté en surface SMC de la figure précédente. Les deux modèles sont disponibles sous forme de rubans, mais la principale différence entre les deux est la façon dont ils sont commandés par le programme. Le modèle NeoPixels n'a besoin que d'un fil pour le signal alors que le DotStar en réclame deux. Vous pourriez en déduire qu'il est plus facile d'utiliser le NeoPixels, mais il pose un vrai problème :

» Le signal envoyé sur cette entrée doit être contrôlé avec une grande précision au niveau

temporel.

» Le Raspberry Pi fonctionne avec un système d'exploitation Linux qui ne permet pas d'espérer une très grande précision temporelle.

Il existe des librairies qui permettent de contourner ce problème, mais chacune apporte ses inconvénients. La librairie Adafruit ne fonctionne qu'avec l'ancien Raspberry Pi modèle 1 ainsi qu'avec le Pi Zéro. La librairie de Pimoroni fonctionne avec tous les modèles, mais elle occupe des ressources matérielles qui servent normalement à la sortie audio sur jack. (Vous pouvez toujours émettre du son sur le connecteur HDMI, mais plus sur le jack audio, car cela provoque des clignotements parasites sur les diodes LED.)

Le modèle DotStar a besoin de deux signaux d'entrée ; il se sert des changements d'état pour décoder les signaux d'entrée. C'est une solution idéale pour une machine fonctionnant sous Linux, car vous ne pouvez jamais savoir à la microseconde près quand la prochaine instruction sera exécutée. Les diodes sont reliées en série, comme dans une guirlande, ce que montre la Figure 17.5. Les signaux sont régénérés dans chaque diode avant d'être transmis à la diode suivante. Autrement dit, il n'y a aucune crainte de voir le signal être dégradé en bout de guirlande.

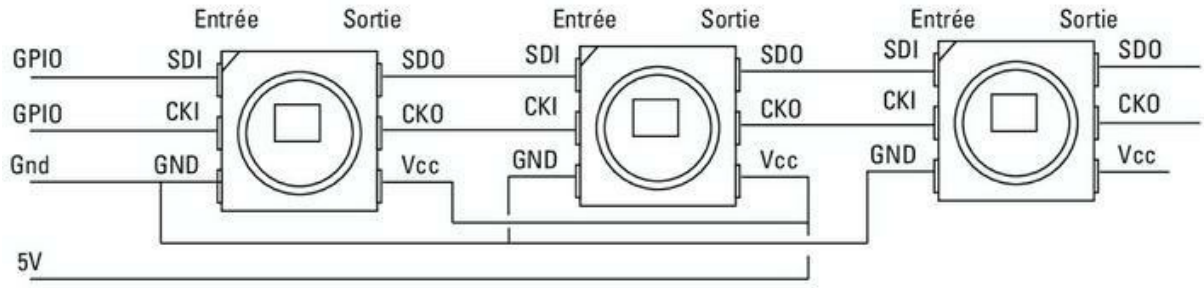


Figure 17.5 : Câblage d'une guirlande de diodes LED APA102.

Dans le modèle APA102, chaque diode possède une entrée Data Input (DI), une sortie Data Output (DO) ainsi qu'une entrée d'horloge Clock In (CI) et une sortie Clock Out (CO). Les deux entrées de la première diode sont tout ce qu'il faut relier à deux broches GPIO. Pour les deux modèles, une fois qu'une diode a reçu sa donnée, les données suivantes sont retransmises à la diode suivante. Une convention a été définie pour distinguer le cas où nous voulons renvoyer de nouvelles données à la première diode LED et non des données pour les lettres suivantes. Cette convention consiste pour le modèle WS2812b à forcer une pause d'émission de longueur supérieure à 50 microsecondes. Pour le modèle

APA102, la convention consiste à envoyer un train d'impulsions de début et de fin.

La Figure 17.6 propose un chronogramme des deux types de LED. Il permet de voir comment les signaux logiques changent pour exprimer un état logique haut (1), un état logique bas (0) ou une fin de transmission.

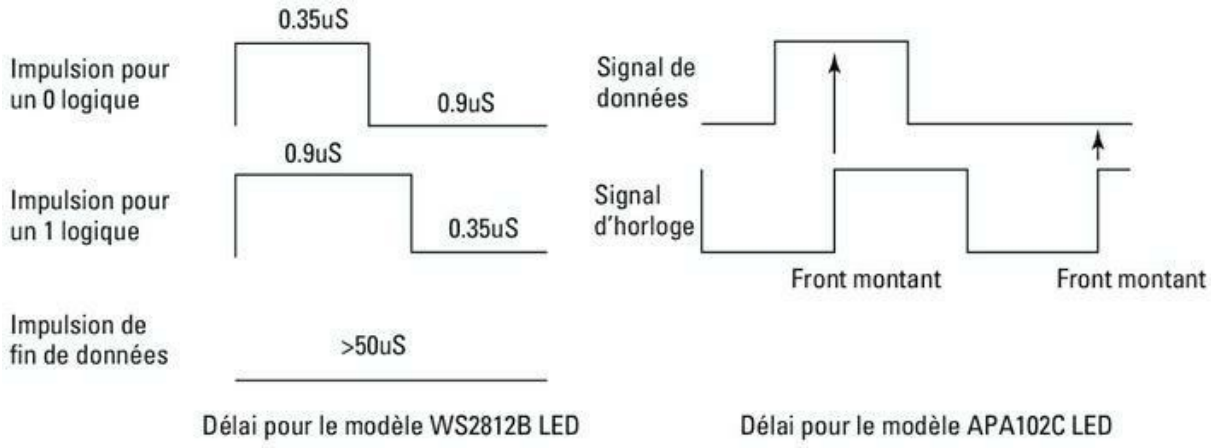


Figure 17.6 : Chronogrammes pour les LED WS2812b et APA102.

Pour le modèle WS2812b, le 1 logique correspond à un état haut pendant 0,9 µs suivi d'un état bas pendant 0,35 µs, et l'inverse pour le 0 logique. Ce couple d'impulsions se répète autant de fois que nécessaire pour alimenter toutes les diodes LED de la guirlande. La fin de transmission est marquée par un état bas pendant au moins 50 µs. Ces durées doivent être respectées à plus ou moins 0,15 µs pour que tout fonctionne.

Dans le cas du modèle APA102, il y a deux signaux qui peuvent être envoyés à une fréquence sélectionnable. Le point essentiel est que lorsque le signal d'horloge passe de zéro à un, c'est-à-dire lors d'un front montant, l'état dans lequel se trouve l'autre signal à ce moment est pris en compte. Cela supprime tout problème de maintien de durée réelle. Il suffit de mettre les broches GPIO dans les bons états.

Il existe un circuit dédié dans le microcontrôleur pour prendre en charge cette génération de signaux. Le protocole correspondant porte le nom SPI (*Serial Peripheral Interface*). Le problème est que ce protocole demande d'utiliser certaines broches dans un mode particulier, contrainte que nous ne voulons pas subir. Pour pouvoir utiliser deux broches quelconques, il faut donc émuler ce protocole, ce que les programmeurs appellent *bit banging*. Voyons comment réaliser cette émulation d'impulsions sur le GPIO.

N. d. T : Pour l'expression *bit banging*, nous parlerons en français d'émulation de protocole.

Émulation du protocole APA102

Chacune des LED d'une guirlande possède un numéro d'ordre, qui est en quelque sorte son adresse. Lorsque nous envoyons des données, nous les transmettons à toutes les diodes, mais nous n'avons besoin de changer les données que pour certaines diodes. Lorsque la couleur d'une diode reste la même, il n'y a rien à changer.

En plus des trois paramètres de luminosité du rouge, du vert et du bleu, le modèle APA102 reconnaît un quatrième paramètre pour la luminosité globale, avec une valeur entre 0 et 31. Ce paramètre contrôle réellement la quantité de courant qui passe dans la diode, comme si la résistance était variable.

En conclusion, chaque diode doit recevoir quatre valeurs numériques : la luminosité générale et les trois composantes primaires. Les données doivent être stockées par le programme dans une zone mémoire spéciale que l'on appelle un tampon ou buffer. En langage Python, ce tampon va prendre la forme d'une liste de données. Une fois que cette liste existe, nous pouvons changer les valeurs qu'elle contient pour faire varier le résultat sur chaque diode LED. Une fois que tous les changements sont faits dans la liste, nous envoyons la totalité du tampon vers la guirlande.

Voyons plus en détail ce qu'il faut stocker dans ce tampon. Le message complet doit commencer par un en-tête, afin de prévenir les diodes LED qu'elles vont recevoir des données. Cet en-tête est représenté par 32 bits à zéro. Il faut donc mettre la broche GPIO de donnée à 0 puis envoyer 32 impulsions sur la broche de l'horloge. Nous pouvons ensuite transmettre les données, un paramètre à la fois. Chaque valeur est sur 32 bits, et chaque bit est présenté sur la broche de sortie juste avant d'amener le signal d'horloge à l'état haut pour qu'il soit pris en compte. Une fois que toutes les diodes ont reçu leurs données, nous devons envoyer le marqueur de fin de message. Il s'agit ici aussi d'une série de bits à zéro, sauf qu'il en faut 32 plus la moitié du nombre de diodes de la guirlande. Le code du Listing 17.2 est un extrait qui montre comment traiter le tampon.

Listing 17.2 : Transmission d'un message APA102 (extrait).

```
io.output(da, io.LOW)          # Broche data a 0
for i in range(0, 32):         # Debut de message
  io.output(ck, io.LOW)        # Broche horloge
```

```
  io.output(ck, io.HIGH)
for i in range(0, nbrLED):    # Envoi data
   d = tabLed[i]             # Paquet de data pour
une LED
   for j in range(0, 32):
     io.output(ck, io.LOW)
     if d & 0x80000000 :
       io.output(da, io.HIGH)
     else:
       io.output(da, io.LOW)
     d = d << 1                      # Decalage
       io.output(ck, io.HIGH)
io.output(da, io.LOW)
for i in range(0, 33+(nbrLED/2) ): # Fin de
message
  io.output(ck, io.LOW)
  io.output(ck, io.HIGH)
```

Nous rappelons qu'il ne s'agit ci-dessus que d'un extrait et non d'un programme complet. Remarquez bien l'utilisation de l'opérateur de décalage vers la gauche **<<** qui sert à déplacer d'une position binaire la donnée pour que le prochain élément à transmettre soit placé dans le bit le plus significatif du mot. C'est la même technique que nous avons utilisée dans le chapitre précédent pour les motifs du dé électronique.

Définition d'une classe

Les instructions d'émulation que nous venons de voir peuvent être utilisées dans tous les projets qui comportent ce genre de diodes LED adressables. Pour simplifier l'utilisation, il est intéressant de réunir ces instructions pour former une vraie classe, afin de pouvoir directement réutiliser ces fonctions dans vos programmes. Il en va un peu de même qu'avec par exemple la librairie RPi. GPIO. Elle contient une définition de classe utilisable dans vos programmes Python. Vous pouvez tout à fait définir vos propres classes et les ajouter au langage, ou, de manière moins ambitieuse, créer le fichier de définition de la classe et le stocker dans le même dossier que le fichier de code source qui a besoin d'utiliser cette classe. Il suffira dans ce dernier de faire référence à la classe. En complément du travail d'émission des données sur les broches GPIO, nous pouvons profiter de cette définition pour ajouter quelques fonctions

complémentaires, comme par exemple le réglage de la luminosité générale, la préparation du tampon de données et le choix d'une couleur initiale pour les diodes LED. Toutes les fonctions qui sont définies pour une classe sont des méthodes, et elles sont appelées comme des fonctions standard. Le Listing 17.3 propose la définition d'une classe d'émulation APA102. Saisissez ou récupérez ce code source et sauvegardez-le dans un fichier portant exactement le nom *apa102bang.py*. Vous stockerez le code source de tous les exemples qui ont besoin de cette classe dans le même dossier ou répertoire.

Listing 17.3 : Définition d'une classe d'émulation APA102.

```
#!/usr/bin/env python3
# Classe de protocole de LED APA102
# Par Mike Cook
import RPi.GPIO as io
io.setwarnings(False)

class Apa102bang(): # Notre classe

  def __init__(self,nLeds,data,clock,lumi):
    self.changerLumigen(lumi)
    self.da = data
    self.ck = clock
    self.nbrLED =  nLeds
    io.setmode(io.BCM)
    io.setup(self.ck,io.OUT)
    io.setup(self.da,io.OUT)
    io.output(self.ck, io.HIGH)
    io.output(self.da, io.HIGH)
    self.tabLed = [self.br<<24 for i in
range(0,self.nbrLED)]

  def changerLumigen(self,lumino):
    if lumino > 31:
       lumino = 31
    if lumino < 0:
       lumino = 0
```

```
    self.br = lumino ¦ 0xE0

  def changerLed(self,pos,col):
    if pos < self.nbrLED and pos >= 0:
       self.tabLed[pos] = (self.br<<24)¦(col[2]
<<16)¦(col[1] <<8)¦col[0]

  def changerTout(self,col):
     for i in range(0,self.nbrLED):
       self.tabLed[i] = (self.br<<24)¦(col[2]
<<16)¦(col[1] <<8)¦col[0]

def afficher(self):
  io.output(self.da, io.LOW)
  for i in range(0,32):                # Debut
    io.output(self.ck, io.LOW)
    io.output(self.ck, io.HIGH)
  for i in range(0,self.nbrLED):
     d = self.tabLed[i]                    # Data
     for j in range(0,32):
       io.output(self.ck, io.LOW)
       if d & 0x80000000 :
         io.output(self.da, io.HIGH)
        else:
         io.output(self.da, io.LOW)
       d = d << 1
       io.output(self.ck, io.HIGH)
  io.output(self.da, io.LOW)
  for i in range(0,33+(self.nbrLED>>1)):    # Fin
io.output(self.ck, io.LOW)
io.output(self.ck, io.HIGH)
```

Vous remarquez que le nom de la classe indiqué après le mot-clé **`class`** ne se distingue du nom du fichier qui la contient que par le passage tout en minuscules dans le fichier. Cette convention est indispensable pour que la classe soit bien reconnue. Chacune des méthodes de la classe est définie avec une instruction **`def`**, comme s'il s'agissait d'une fonction classique.

La classe est initialisée par la fonction spéciale **`__init__`**. On appelle également cela le constructeur de la classe. Vous constatez également qu'un grand nombre de variables sont précédées du préfixe **`self.`**, ce qui informe le compilateur que la variable est en réalité une donnée membre de la classe qui peut prendre des valeurs différentes selon l'objet qui a été créé à partir de cette classe.

Les noms des méthodes permettent en général de savoir à quoi elles servent. Par exemple, la méthode **`changerLumigen()`** se contente de préparer une variable qui va contenir la luminosité générale qui sera appliquée lors des prochains transferts de paramètres vers les diodes. Elle n'a pas d'effet sur la luminosité actuelle. Si vous avez besoin de changer la luminosité générale actuelle, il suffit de modifier la valeur correspondante dans la liste des paramètres des diodes avant d'appeler la méthode qui écrit les données.

Pour tester cette nouvelle classe, nous allons utiliser une barrette de diodes LED de Pimoroni, constituée de huit diodes du modèle APA102C. La barrette est soudée sur un connecteur femelle à 40 broches, qui vient directement s'enficher sur le connecteur de votre carte RasPi. En réalité, la barrette n'utilise que deux broches GPIO en plus des deux broches d'alimentation 5 V et masse.

N. d. T. : Pour insérer la barrette Blinkt ! dans le connecteur GPIO, les deux coins arrondis de la barrette doivent être tournés vers l'extérieur de la carte RasPi.

Le Listing 17.4 est un programme de test de la classe. Récupérez ou saisissez ce code source et stockez-le dans le même dossier que celui contenant le fichier de la classe.

Listing 17.4 : Programme de test de la classe APA102.

```
#!/usr/bin/env python3
# Demo de la classe Apa102bang

import time
from apa102bang import Apa102bang

longueur = 8
lumi     = 4
dataPin  = 23
clockPin = 24      # Broches Blinkt!
```

```
barreLed =
Apa102bang(longueur,dataPin,clockPin,lumi)

def main():
  print("Demo APA102 - Ctrl C pour quitter")
  while True:
    for i in range(0,8):
      barreLed.changerLed(i,(120,0,0)) # rouge
      barreLed.afficher()
      time.sleep(0.06)
    time.sleep(0.6)
    for i in range(0,4):
      barreLed.changerLed(i,(0,120,0)) # vert
      barreLed.afficher()
      time.sleep(0.06)
    time.sleep(0.6)
    for i in range(4,8):
      barreLed.changerLed(i,(0,0,120)) # bleu
      barreLed.afficher()
      time.sleep(0.06)
    time.sleep(0.6)
    for i in range(0,8):
      barreLed.changerLed(i,(0,0,0))   # noir
      barreLed.afficher()
      time.sleep(0.06)
    time.sleep(0.6)

# Main program logic:
if __name__ == '__main__':
    main()
```

Le programme commence par définir le nombre de diodes LED de la guirlande, la luminosité générale et les deux broches pour les données et l'horloge, ces numéros étant figés par construction du composant Blinkt ! (vous pourriez utiliser n'importe quelles autres broches GPIO compatibles, notamment pour utiliser plusieurs guirlandes). La variable nommée `barreLed` va incarner l'objet créé par instanciation de la

classe. Vous appelez ensuite les méthodes de la classe en indiquant ce nom d'objet en préfixe. Ce programme de test se limite à afficher tour à tour les trois couleurs primaires pour chaque diode puis à revenir au noir. Dans la méthode **`changerLed()`**, vous remarquez que nous utilisons les structures de données dites *tuples* pour stocker les trois couleurs. Le contenu du tampon de données n'est transmis à la guirlande que dans la méthode **`afficher()`**. Si vous oubliez d'appeler cette méthode, les diodes ne vont jamais changer de couleur.

Tout cela est fort intéressant, mais nous pourrions sans doute mieux tirer profit de notre guirlande. Voyons si nous pouvons faire un petit jeu vidéo minimaliste.

Les sept envahisseurs de l'arc-en-ciel

Le jeu que nous vous proposons, *Rainbow Invaders*, s'inspire librement d'un ancien jeu d'arcade, *Space Invaders*. La différence est qu'il ne fonctionne que sur une colonne. Les envahisseurs larguent une par une des bombes dont la couleur varie d'une bombe à l'autre, et c'est à vous de détruire ces bombes en plein vol en envoyant un rayon laser de la même couleur que la bombe. Vous avez trois boutons pour choisir la couleur. Cela semble simple non ? Pas si vite. Les envahisseurs disposent de sept couleurs de bombes différentes : les trois couleurs primaires, les trois couleurs secondaires et le blanc. Autrement dit, vous devez trouver la bonne combinaison entre les trois boutons pour générer la bonne nuance de rayon laser. Lorsque vous avez été touché par trois bombes, votre base est détruite. La couleur de la base change en fonction de son état de préservation. Plus vous réussissez à détruire une bombe tôt pendant sa chute, plus votre score augmente. En ajoutant quelques effets sonores, nous obtenons un petit jeu.

Avant d'envisager la programmation du jeu, il nous faut réaliser le montage du circuit. Nous avons ajouté trois boutons-poussoirs, mais il y a un petit problème. La barrette de diodes Blinkt ! vient coiffer entièrement le connecteur GPIO, et empêche donc d'accéder aux autres broches en dehors des quatre qu'elle utilise. Il y a plusieurs manières de résoudre le problème. Vous pouvez souder quatre fils à l'arrière de la barrette et les connecter au connecteur GPIO par fils volants avec des embouts femelles. C'est ce que nous avons fait dans la Figure 17.7. Vous pouvez également vous procurer une carte d'extension comme la Black HAT Hack3R ou la mini Black HAT Hack3R, ce que nous avons fait dans la Figure 17.8. Vous pouvez enfin brancher quatre straps dans le connecteur GPIO

femelle de la barrette Blinkt !, ce qui vous laisse toute latitude pour connecter les fils des trois boutons sur le connecteur GPIO.

Votre base de tir dans la barrette de LED correspond à la diode LED de droite, mais avec la carte d'extension ou les straps volants, vous pouvez repositionner la barrette pour que le jeu soit vertical.

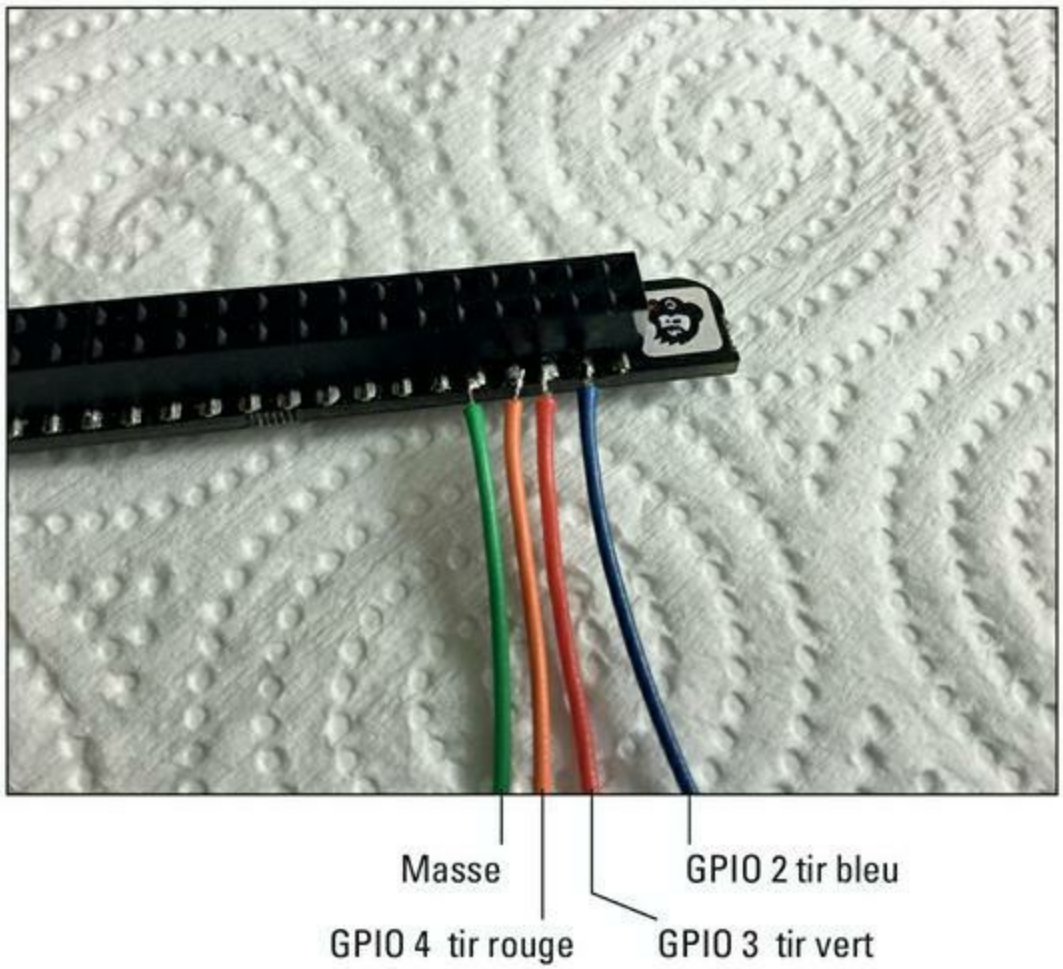


Figure 17.7 : Soudure de quatre straps à l'arrière de la barrette de LED.

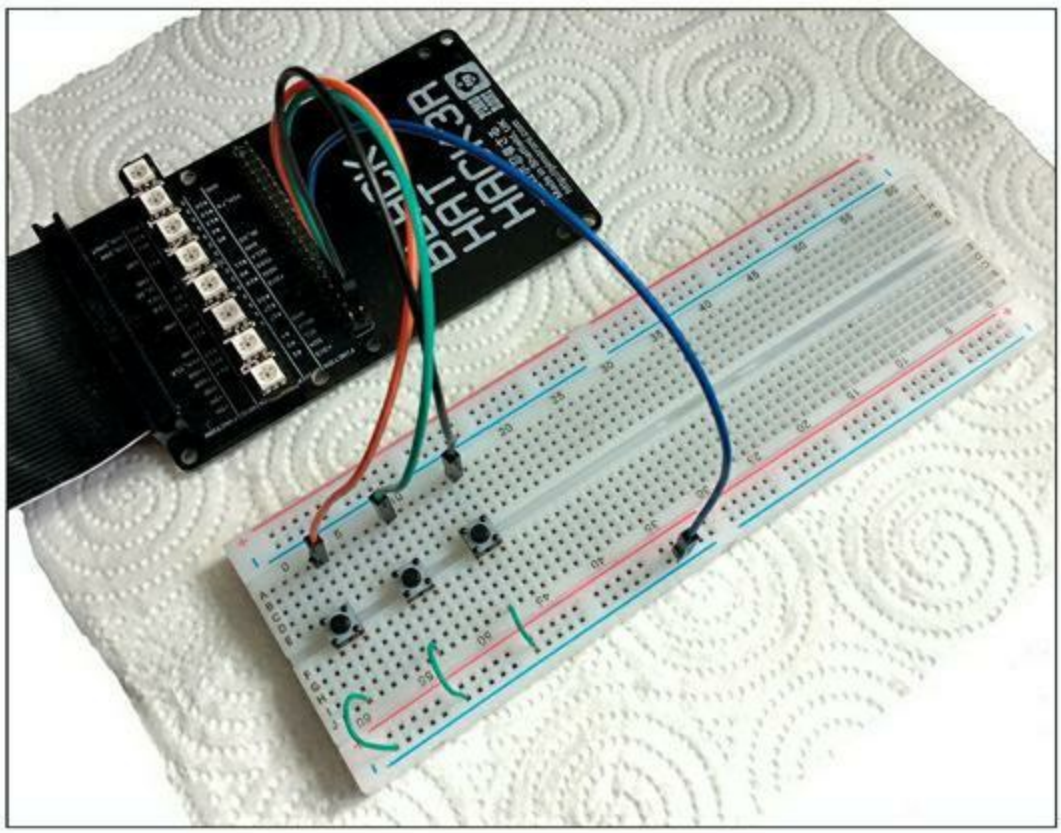


Figure 17.8 : Carte d'extension Black HAT Hack3R pour retrouver l'accès aux broches GPIO.

Pour le programme, nous avons tiré profit de la librairie Pygame qui est préinstallée, car elle simplifie l'exploitation des effets sonores. Vous devez stocker vos fichiers de sons dans un sous-répertoire de celui contenant le programme et qui portera le nom *sounds*. Pour les sons, récupérez ceux des exemples du langage Scratch. Accédez au répertoire suivant :

```
/usr/share/scratch/Media/Sounds
```

Descendez ensuite dans le sous-répertoire *Electronic*. Copiez dans *sounds* un par un les trois fichiers de son *ComputerBeeps2*, *Laser1* et *Screech*. Faites de même pour le son nommé *Pop* du sous-répertoire *Effects*.

Découvrons maintenant le code source du jeu (Listing 17.5).

Cet exemple utilise bien sûr la version française de la définition de la classe, donc le fichier *apa102bangfr.py*.

Listing 17.5 : Le jeu Rainbow Invaders.

```
#!/usr/bin/env python3
```

```
# Rainbow Invaders by Mike Cook
# VF par O.E.

import RPi.GPIO as io
import pygame
import time, random
from apa102bangfr import Apa102bangfr

def main():
  global rayon, encore, altiBmb, coulBmb
  print("Rainbow Invaders FR")
  init()
  encore = True
  monScore = 0  # bombHit = False
  altiBmb = 8
  coups = 3
  ciel.changerLed(0, base[coups])
  ciel.afficher()
  time.sleep(vitesseJ)
  tBmbSuiv = time.time() + vitesseJ
  coulBmb = (0,0,0)
  colorerBmb()
  while 1:
    while encore:
      if time.time() > tBmbSuiv:
         altiBmb -=1
         voirBombe(altiBmb)
         tBmbSuiv = time.time() + vitesseJ
      monScore = tirer(altiBmb, monScore)
      if altiBmb == 0 :
          ciel.changerTout((0,0,0))
          ciel.changerLed(0,(255,0,0))
          ciel.afficher()  # explosion
          sonJ[3].play()   # coup base
          time.sleep(2.5)  # temps d'appui
        coups -= 1
```

```
            if coups == 0:
             encore = False
             else:
             print("Base:-",coups,"avant
destruction")
             ciel.changerLed(0,base[coups])
             colorerBmb()
             altiBmb = 8
       ciel.afficher()
     print("Base disparue:- votre score",monScore)
     time.sleep(vitesseJ * 10)
     print("Appuyez sur un bouton pour rejouer!")
     while io.input(feuR) and io.input(feuV) and
io.input(feuB):
         pass
     print("Nouvelle partie...")
     coups = 3
     encore = True
     altiBmb = 8
     ciel.changerLed(0, base[coups])
     ciel.afficher()
     colorerBmb()
     while (not io.input(feuR)) ¦ (not
io.input(feuV)) ¦
(not io.input(feuB)):
         pass

def lireCTir():
     m = 128
     b = ((not io.input(feuR))*m,(not
io.input(feuV))*m,
(not io.input(feuB))*m)
     return b

def voirLaser(col,limit):
    for i in range(1,limit):
```

```
      ciel.changerLed(i,col)
      ciel.afficher()
      time.sleep(0.05)

def voirBombe(pos):
   global coulBmb
   voirLaser((0,0,0),8)
   ciel.changerLed(pos,coulBmb)
   ciel.afficher()

def tirer(alti, score):
   global rayon,coulBmb,altiBmb
   if lireCTir() != rayon:    # laser
     rayon = lireCTir()
     if rayon != (0,0,0):
sonJ[1].play()
     voirLaser(rayon, alti)
    if rayon == coulBmb :   # au but
       time.sleep(0.2)
       sonJ[2].play()        # Son impact
       score += alti
       time.sleep(1.5)       # Attendre fin du
tir
       altiBmb = 8
       colorerBmb()
   return score

def colorerBmb():
   global coulBmb
   sonJ[0].play() # bomb incoming
   while 1  :
     r = random.randint(0,1) * 128
     g = random.randint(0,1) * 128
     b = random.randint(0,1) * 128
     if not(r==0 and g==0 and b==0) and coulBmb
!= (r,g,b) :
```

```
            break
    coulBmb = (r,g,b)

def init():
    global
feuR,feuV,feuB,rayon,ciel,vitesseJ,base,sonJ
    random.seed()
    io.setmode(io.BCM)
    feuR = 4
    feuV = 3
    feuB = 2
    io.setup([feuR,feuV,feuB],io.IN,
pull_up_down=io.PUD_UP)
    rayon = lireCTir()

    guirla = 8
    lumino = 4
    broData  = 23       # Broches pour Blinkt
    broClock = 24
    ciel =
Apa102bangfr(guirla,broData,broClock,lumino)
    vitesseJ = 0.3     # Delai en secondes
    base = [(0,0,0),(32,0,32),(0,64,0),
(0,128,128)]   # Couleurs de base

    pygame.mixer.quit()
    pygame.mixer.init(frequency=22050, size=-16,
channels=2, buffer=512)
    discot =
["ComputerBeeps2","Laser1","Pop","Screech"]
    sonJ = [
pygame.mixer.Sound("sounds/"+discot[sound]+".wav")

                    for sound in range(0,4)]

#
```

```
if __name__ == '__main__':
  try:
    main()
  except KeyboardInterrupt:
   pass
# turn off all LEDs
ciel.changerTout((0,0,0))
ciel.afficher()
```

La fonction principale **`main()`** est constituée de deux grandes boucles de répétition ; c'est ici que se passe l'essentiel de l'action. Le programme reste dans la boucle interne contrôlée par **`while encore`** pendant toute une partie. Des bombes sont lâchées jusqu'à ce que trois aient touché la base, c'est-à-dire atteint l'altitude zéro. Le premier test **`if`** vérifie que le délai minimal pour faire descendre la bombe d'une case est écoulé. L'appel à la méthode **`tirer()`** teste l'appui sur les boutons pour générer un tir. Si c'est le cas, le laser est déclenché en même temps qu'un son. Si la couleur du laser correspond à celle de la bombe, nous déclenchons le son d'explosion et la hauteur actuelle de la bombe est ajoutée au score. Nous ramenons ensuite la hauteur de la bombe à huit puis nous générons une nouvelle couleur de bombe et le programme refait un tour.

Si l'altitude de la bombe est égale à zéro, nous colorions la base en rouge pour marquer l'impact et émettons un son d'explosion. Nous enlevons un au nombre de tirs possibles. Si la valeur atteint zéro, la partie est finie et nous forçons la variable `encore` à la valeur False, ce qui fait sortir de la boucle. Nous arrivons alors aux dernières lignes de la grande boucle **`while 1`** pour afficher le score, puis nous placer en attente d'appui sur un bouton pour démarrer une nouvelle partie.

L'objet qui incarne la guirlande de LED porte le nom `ciel`, ce qui est suffisamment descriptif. La vitesse du jeu est contrôlée par la variable `vitesseJ` qui est définie dans la fonction **`init()`**. Lorsque vous quittez le programme par les deux touches Ctrl + C, ce qui correspond à une interruption clavier, n'oubliez pas d'éteindre toutes les diodes pour arrêter le jeu proprement.

Vous pouvez rendre le jeu plus facile en jouant sur la vitesse et le nombre de coups au but que vous pouvez encaisser ou le rendre plus difficile en ajoutant par exemple un quatrième bouton pour vous obliger à gérer deux niveaux de luminosité. Dans ce cas, vous ne pourrez détruire une bombe

qu'en trouvant la bonne nuance de couleur parmi sept et la bonne luminosité. Dans ce cas, il faut également modifier la génération des bombes pour alterner entre les deux brillances. Cet exercice n'est pas très difficile, mais il faut réfléchir un peu.

Le pied jongleur *Keepy Uppy*

Nous pouvons créer un autre jeu sans rien changer au montage électronique. Ce jeu, que nous appelons *Keepy Uppy*, consiste à simuler le jonglage avec un ballon sur un pied, comme le font dans le monde entier tous les enfants qui rêvent de devenir des stars du ballon rond. Le ballon est symbolisé par une diode LED qui monte et descend. À la différence du jeu précédent, la couleur est fixe et vous n'avez besoin d'appuyer que sur un seul bouton pour faire rebondir le ballon lorsqu'il touche votre pied, c'est-à-dire lorsqu'il atteint la diode LED du bas. Si vous appuyez au bon moment, le ballon remonte jusqu'en haut puis redescend. Pour rendre le jonglage plus intéressant, la vitesse du jeu augmente lorsque vous appuyez dans la seconde moitié du temps autorisé pour frapper le ballon, ce qui fait qu'il est ensuite plus difficile de frapper juste au moment où le ballon le touche. Si vous réussissez à le frapper dans la première moitié du temps autorisé, le jeu revient à sa vitesse initiale. Cette astuce augmente un peu la difficulté. Découvrons donc le code source de ce jeu, dans le Listing 17.6.

Comme le précédent, cet exemple utilise la version française de la définition de la classe, donc le fichier *apa102bangfr.py*.

Listing 17.6 : Le jongleur de ballon *Keepy Uppy*.

```
#!/usr/bin/env python3
# KeepeeUpee by Mike Cook

import RPi.GPIO as io
import time, random
from apa102bangfr import Apa102bangfr

nBarette = 8

def main():
```

```
  global encore,altiBal,coulBal
  print("Keepee Upee FR")
  init()
  encore = True
  sensBallon = -1
  altiBal = nBarette
  pied = 0
  score = 0
  vitesseK = vitesseJ
  plato.changerLed(0,coulPied[pied])
  plato.afficher()
  time.sleep(vitesseJ)
  mouvBalSuiv = time.time()+vitesseJ
  kickPrec = not(io.input(broKick))
  while 1:
    while encore:
      kickNouv = not(io.input(broKick))
      if kickNouv and not(kickPrec) :
        if altiBal == 0: #   kick
          sensBallon = sensBallon * -1
          # Accelerer si en retard
          if mouvBalSuiv - time.time() <
vitesseK/2 :
           vitesseK = vitesseK / 2
            else:
            vitesseK = vitesseJ
           mouvBalSuiv = time.time()-1.0
           pied = 1
           score += 1 # Augmenter score
         else:
          pied = 2
      if time.time() > mouvBalSuiv:
         plato.changerLed(altiBal,(0,0,0))
         altiBal += sensBallon
         plato.changerLed(0,coulPied[pied])
         plato.changerLed(altiBal,coulBal)
```

```
            pied = 0
            mouvBalSuiv = time.time()+vitesseK
         if altiBal >= nBarette-1 :
            sensBallon = -1
         if altiBal < 0: # Ballon manque
            encore = False
         kickPrec = kickNouv
         plato.afficher()
      print("Ballon par terre. Votre score:",score)
      while not(io.input(broKick)):
          pass
      time.sleep(vitesseJ* 5)
      print("Appuyez pour rejouer")
      while not(io.input(broKick)):
          pass
      score = 0
      encore = True
      altiBal = nBarette
      sensBallon = -1
      plato.afficher()
      vitesseK = vitesseJ
      while io.input(broKick):
          pass

def init():
   global broKick,plato,vitesseJ,coulPied,coulBal
   random.seed()
   io.setmode(io.BCM)
   broKick = 4
   io.setup(broKick,io.IN,
pull_up_down=io.PUD_UP)
   lumi = 4
   dataPin  = 23
   clockPin = 24 # Blinkit
   plato =
Apa102bangfr(nBarette,dataPin,clockPin,lumi)
```

```
    vitesseJ = 0.3
    coulBal  = (128,128,0)
   coulPied = [(32,32,32),(128,0,0),(0,0,128)]

# Main program logic:
if __name__ == '__main__':
  try:
    main()
  except KeyboardInterrupt:
    pass
# turn off all LEDs
plato.changerTout((0,0,0))
plato.afficher()
```

Ce programme ressemble beaucoup au précédent dans la mesure où l'essentiel du travail est réalisé dans la boucle de répétition **`while`** de la fonction principale. Un point particulier qu'il a fallu résoudre est celui-ci : le joueur pourrait se contenter de maintenir enfoncé le bouton en permanence, ce qui renverrait le ballon toujours au bon moment. Pour empêcher cette simplification, nous codons un détecteur de changement d'état du bouton. Il consiste à mémoriser l'état actuel du bouton puis à le comparer à son état dans le tour de boucle précédent. Nous ne validons le jonglage que si le bouton est enfoncé alors qu'il ne l'était pas précédemment. Cela correspond à une détection de changement d'état. Sur un chronogramme, c'est l'équivalent d'un front montant.

Chaque jonglage réussi augmente votre score de 1, comme en vrai. La variable nommée `mouvBalSuiv` contient le temps du prochain déplacement du ballon. Nous pouvons savoir à quel moment le pied a frappé le ballon en soustrayant à ce délai l'heure actuelle, ce qui permet de savoir si le ballon a été frappé au bon moment et de décider d'accélérer le jeu ou de revenir à la vitesse initiale.

Pour rendre ce jeu plus aérien, il suffit d'utiliser une guirlande plus longue que celle de huit diodes de l'exemple. Il suffit de changer la valeur de la variable `nBarette` tout au début du code source. Comme nous allons le voir, nous pouvons utiliser des guirlandes bien plus longues que la simple barrette Blinkt ! .

Des galaxies de LED

L’énorme avantage des guirlandes de diodes LED du type APA102 est cette possibilité de contrôler un grand nombre de diodes avec deux fils. Mais avant d’envisager une guirlande de centaines de diodes, voyons quelles sont les limites pratiques, et comment en contourner certaines.

Limitations en courant

Lorsque vous tentez de contrôler un grand nombre de diodes LED, c’est le courant consommé qui est votre première limitation. Une seule diode consomme très peu, mais tout petit nombre multiplié finit par produire une valeur importante. Supposons que vous utilisiez votre carte RasPi avec une alimentation supportant 2 A, soit quasiment le double de ce que consomme la carte elle-même. Il faut toujours prendre une marge de sécurité pour une alimentation. Nous avons considéré que l’alimentation pouvait fournir 80 % de sa valeur théorique, soit 1,6 A. En conséquence, il nous reste environ 600 mA disponibles pour les diodes. Si chaque diode consomme 60 mA (20 mA par couleur), nous pouvons contrôler au maximum 10 diodes LED, en les allumant en blanc à pleine puissance.

La bonne nouvelle est que le rendement des diodes LED APA102 est tel qu’il est inutile de les exploiter à leur luminosité maximale. En réduisant la luminosité générale, vous pouvez donc contrôler un plus grand nombre de diodes. Vous pouvez bien sûr réduire la luminosité en jouant sur le rapport cyclique des signaux PWM. Au lieu de demander la valeur maximale 255 pour le rouge, vous vous limitez à 128, ce qui consomme deux fois moins de courant.

Un danger existe cependant, car on fait aisément des erreurs de programmation, ce qui pourrait par exemple envoyer des valeurs de luminosité trop importantes dans le signal PWM. Ce signal ne peut en aucun cas limiter le courant qui traverse les diodes LED, car il n’a d’effet que sur le courant moyen. À titre d’information, sachez que le signal PWM permet de moduler par exemple 20 mA de courant par étapes de 78 µA (il permet en effet 256 valeurs différentes entre zéro et un.)

Une technique beaucoup plus sûre pour contrôler la luminosité consiste à utiliser le paramètre de luminosité générale dont sont dotées les diodes APA102. Rappelons que ce paramètre vient s’ajouter aux trois paramètres des couleurs primaires dans le signal PWM. Pour l’instant, nous nous sommes limités à régler de façon fixe la luminosité générale. C’est une bonne technique pour éviter toute erreur de programmation. Ce paramètre général contrôle réellement le courant qui passe par la diode et il offre 31 niveaux différents. Si une diode LED peut consommer au maximum 60 mA, avec une luminosité générale de 8, elle ne consommera plus que 15,48 mA. Les 600 mA que permet notre alimentation d’exemple

suffisent donc à 38 diodes. De plus, ce niveau de luminosité 8 est encore très brillant, surtout si vous regardez directement les diodes. Vous pouvez descendre jusqu'à 2, et ainsi alimenter jusqu'à 155 diodes LED tout en gardant un effet intéressant.

En revanche, si vous allez au-delà de cette limite, vous devrez vous équiper d'une alimentation indépendante pour la guirlande. Sa mise en place est très simple : il suffit d'alimenter les deux broches d'entrée 5 V et masse de la guirlande, et de bien vérifier que vous avez interconnecté la masse de cette alimentation à celle de la carte RasPi.

Discrimination des états logiques

Deux autres limitations sont à prendre en compte pour une guirlande de diodes LED : les niveaux des signaux et la quantité de mémoire. Au niveau de la mémoire, vous n'avez aucun souci à vous faire avec votre carte RasPi (ce qui n'est pas le cas des autres cartes microcontrôleurs telles que les Arduino). En revanche, vous devez vous inquiéter de la bonne détection des états logiques hauts du signal. Dans tous les exemples de ce chapitre, nous avons relié la barrette directement aux broches de sortie GPIO. Pourtant, les diodes LED de cette barrette sont prévues pour être alimentées en 5 V ; elles s'attendent donc à recevoir des signaux de commande logique avec 5 V pour le niveau haut.

La spécification de ces composants édicte que la tension minimale qui va être considérée comme un état logique haut doit être au minimum de 0,7 fois la tension d'alimentation. Pour une alimentation 5 V, l'état logique haut commence donc à partir de 3,5 V, ce qui est un peu au-dessus des 3,3 V que peut fournir une broche GPIO au maximum. Étonnamment, cela semble fonctionner malgré tout, mais rien ne garantit que cela reste le cas dans toutes les conditions de l'environnement (parasites et température). Voilà pourquoi on prévoit souvent un circuit pilote pour remonter les tensions des deux signaux de l'horloge et des données jusqu'à 5 V. Cette précaution devient indispensable lorsque l'on augmente la longueur de la guirlande, d'autant que les interférences électromagnétiques de l'environnement, telles que celles des moteurs ou des tubes néons, prennent alors de plus en plus d'importance. La Figure 17.9 montre un schéma de circuit pilote que l'on peut réaliser avec un composant 74LS14 ou 74AHCT14.

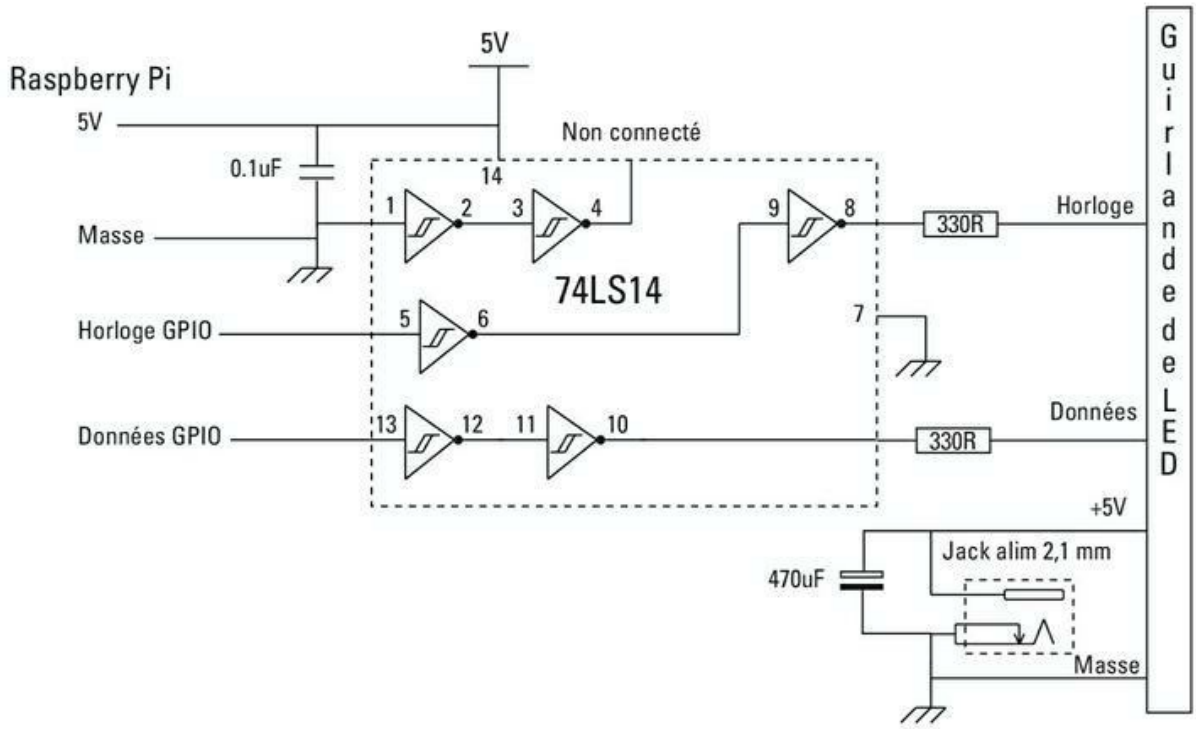


Figure 17.9 : Circuit pilote pour APA102.

Par construction, chacun des circuits élémentaires de ce composant (chaque triangle) est inverseur (les un deviennent des zéros). Comme nous ne voulons pas inverser les états, nous faisons transiter chacun des deux signaux par deux portes successives pour revenir dans le sens original. Nous disposons en sortie d'une tension de 5 V. La résistance en série protège le circuit en cas de chute de la tension d'alimentation. Vous trouverez ce genre de circuit pilote précâblé autour d'un composant 74HTC125. Notez bien la présence du gros condensateur au niveau de l'alimentation. Vous pouvez choisir une valeur encore supérieure pour encore mieux filtrer cette tension.

Fréquence de rafraîchissement

Il nous reste à considérer le problème de la fréquence maximale à laquelle nous pouvons faire évoluer le motif des diodes LED. En utilisant la technique d'émulation de protocole décrite dans ce chapitre, il faut entre 0,45 et 0,23 ms pour envoyer des données vers une seule diode. Cette variation dans le délai vient du fait que le système Linux préempte des tranches de temps quand il en a besoin. Le rafraîchissement ne va donc se produire qu'au moment où le système sera prêt à redonner le contrôle à votre programme. Si une interruption se produit au moment où vous êtes en train d'envoyer les bits vers la guirlande, la durée du rafraîchissement en sera allongée. En pratique, pour 144 diodes LED, vous pouvez rafraîchir le motif environ 15 fois par seconde, au pire.

Rappelons qu'une fois que le rafraîchissement est fait, vous n'avez plus besoin d'agir au niveau de votre programme, jusqu'à la prochaine mise à jour du motif.

Autres modèles de guirlandes LED

Vous trouverez de nombreux formats de guirlandes LED, les plus intéressants étant sans doute les LED programmables de la série RasPiO InsPiRing. Vous en trouverez de formes très diverses, barres, cercles, triangles et carrés. Vous trouverez même un circuit pilote pour générer les bonnes tensions. Ce circuit comporte une prise pour y ajouter un circuit de conversion analogique vers numérique. Cela vous permet de mesurer une tension analogique, par exemple provenant d'un bouton rotatif ou d'un capteur. La Figure 17.10 montre une guirlande triangulaire et une barrette.

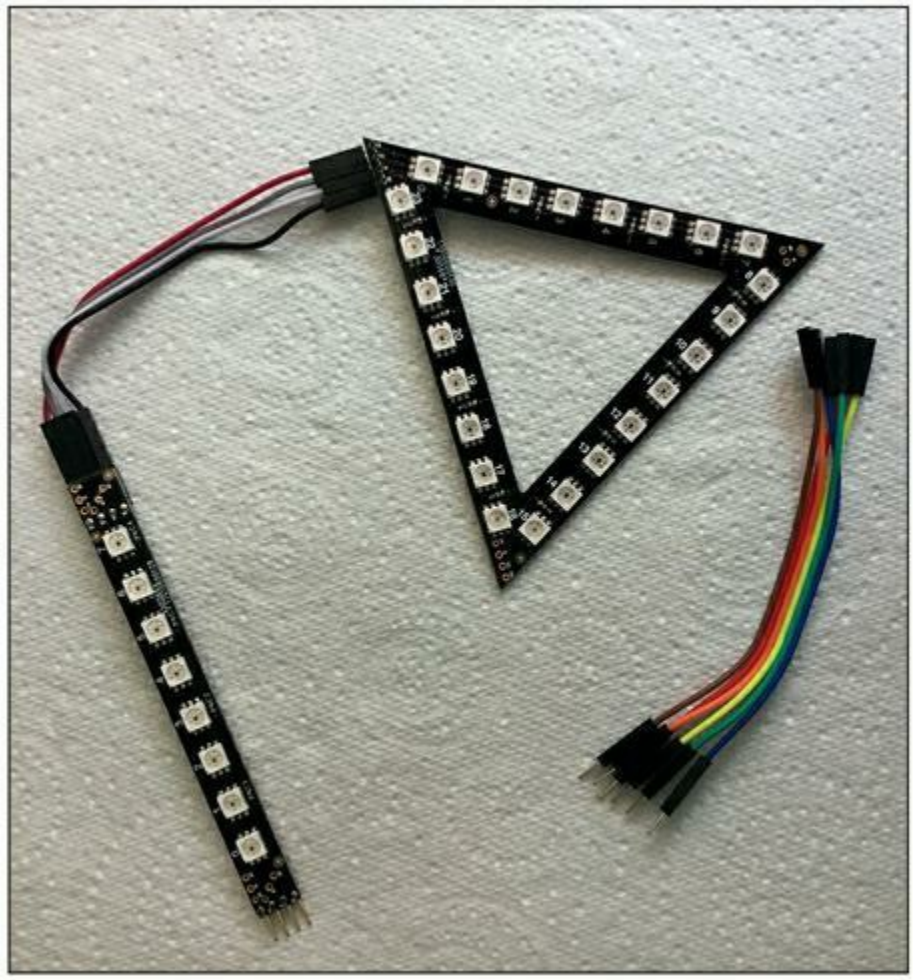


Figure 17.10 : Deux formats de guirlandes LED de la série RasPiO InsPiRing.

Ces composants sont très faciles à utiliser, car chacun est doté d'une prise mâle et d'une autre femelle pour l'entrée et pour la sortie, ce qui permet facilement de prolonger une guirlande avec une autre. Nous adorons celle

en forme de triangle. Vous pouvez facilement construire une pyramide de diodes LED 3D avec. Notez que ces composants fonctionnent non seulement avec votre carte Raspberry Pi, mais avec bien d'autres modèles de microcontrôleurs.

Rubans de LED

Un autre format très répandu, notamment en décoration intérieure, est le format ruban. Les diodes LED sont installées sur un circuit souple, et vendu en bobines de 0,5 à 4 m de long. Vérifiez que vous achetez bien le modèle APA102 et non le modèle WS2812b qui est plus économique. Vous serez ainsi certain que les exemples de ce chapitre fonctionneront. Tenez compte également de la densité de diodes au mètre qui peut aller de 30 à 144. Le prix du ruban dépend bien sûr de cette densité. Il existe plusieurs variantes au niveau de la présentation : le support peut être blanc ou noir, recouvert de silicone ou enchâssé dans un tube étanche. Vous pouvez facilement recouper ce genre de ruban à la bonne longueur, avec un ciseau bien aiguisé.

Les rubans APA102 existent aussi avec des diodes LED blanches au lieu de RGB. Ils sont très utilisés pour l'éclairage intérieur, notamment dans les cuisines. La luminosité est facilement réglable.

Matrices de LED

Un autre format répandu est le format matrice de LED, constitué de lignes et de colonnes. La société Adafruit propose un grand nombre de références dans ce domaine, et notamment un disque géant de 24 cm de diamètre. Parmi les formats plus communs, on trouve des matrices de 8 sur 32, 16 sur 16 ou 8 sur 8, sur support souple. Notre préférence va à la matrice haute densité 8 sur 8 sur support rigide. Elle embarque des minidiodes de 1,8 mm de côté, ce qui permet de faire tenir les 64 diodes sur une plaque de 2,5 cm de côté. La consommation maximale de chaque diode est de 40 mA, ce qui est inférieur à celle des diodes de taille normale. Le courant maximal peut cependant monter jusqu'à 2,5 A, ce qui fait un peu chauffer le circuit. Cela dit, le rendement lumineux est tel que vous pouvez réduire de beaucoup la luminosité. Il faut prévoir un peu de soudure pour relier les fils et le condensateur à l'arrière du circuit, mais tout est indiqué. Nous avons reconnecté les sorties d'alimentation 5 V et masse sur les entrées correspondantes, pour profiter d'une meilleure distribution de l'énergie. Le résultat est visible dans la Figure 17.11.

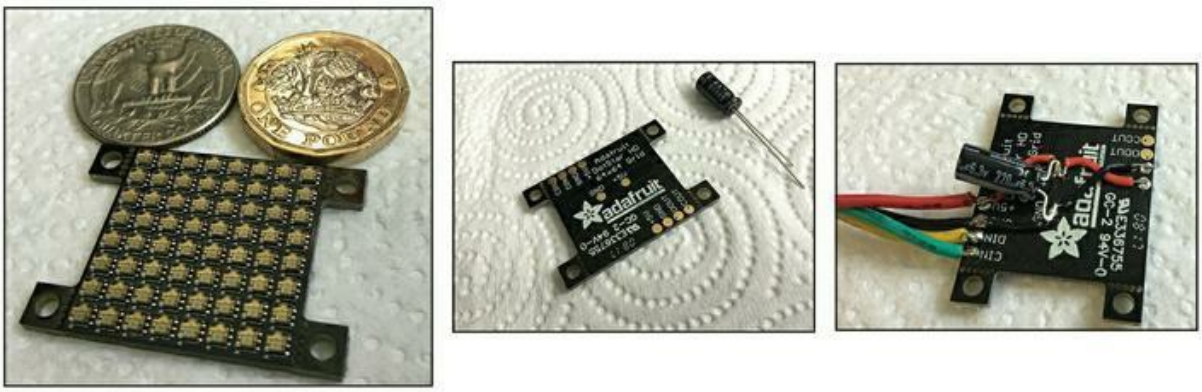


Figure 17.11 : Matrice de LED Adafruit haute densité.

La référence à chercher pour obtenir cette matrice chez Adafruit est *DotStar High Density 8x8 Grid*.

Ce genre de matrice peut être utilisée dans de nombreuses applications, par exemple en tant que broche décorative à porter. Pour réaliser ce genre de projet, il nous a fallu imaginer des motifs colorés dynamiques et attrayants.

De nos jours, les langages de programmation sont tous dotés d'une batterie de fonctions élémentaires d'affichage graphique, c'est-à-dire des primitives qui permettent de dessiner des lignes et des figures. Mais lorsqu'il s'agit de travailler avec une matrice d'affichage, il faut écrire ces primitives soi-même.

Dans un ruban de LED, chacune des diodes est identifiée entre zéro et la longueur du ruban. L'identification des diodes dans une matrice se base sur le même principe. Il est cependant plus efficace de considérer la matrice comme un tableau avec des lignes X et des colonnes Y. Une des fonctions élémentaires dont nous aurons besoin va servir à convertir un couple de coordonnées X, Y en un numéro de diode. Le détail de la conversion diffère selon le mode de câblage des diodes dans la matrice.

Les matrices proposées par Adafruit sont fabriquées selon la convention « Montant départ gauche » (*Row bottom-up*). La première diode, de numéro zéro, se trouve dans l'angle inférieur gauche de la matrice et les numéros augmentent horizontalement vers la droite. À la fin de la ligne du bas, on revient au début de la ligne suivante pour continuer la numérotation. Cela se poursuit jusqu'à arriver en haut à droite. L'autre convention de câblage correspond au serpentin, dans lequel on alterne entre numérotation de gauche à droite et de droite à gauche. Les deux conventions sont présentées dans la Figure 17.12.

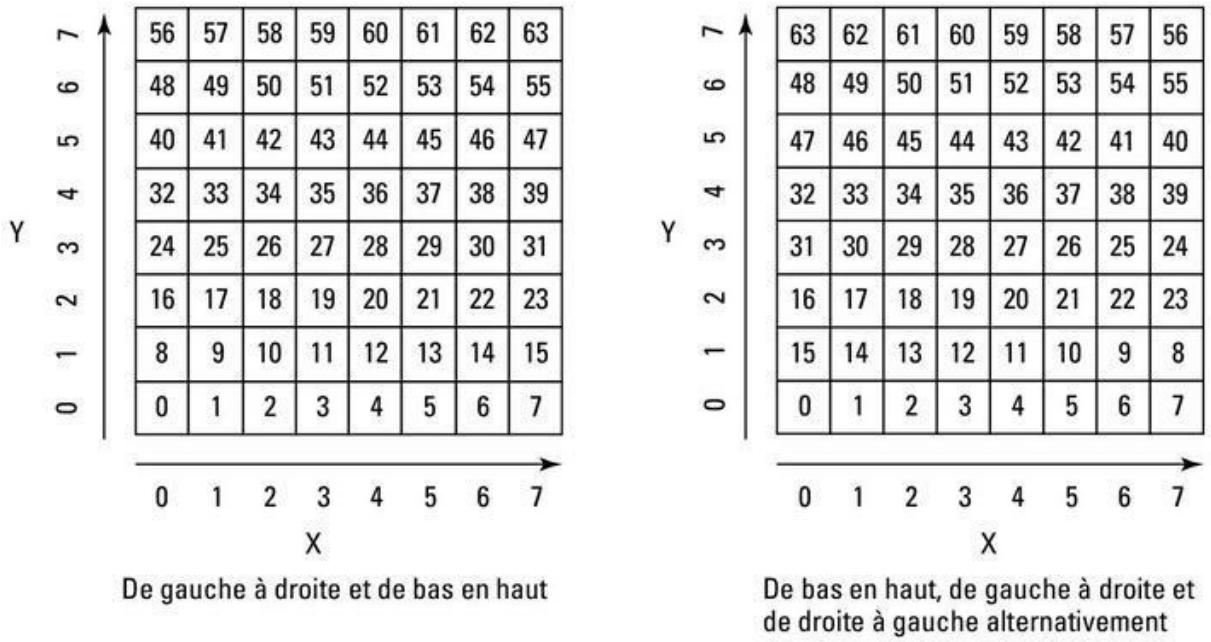


Figure 17.12 : Les deux conventions d'adressage des diodes d'une matrice.

Pour obtenir le numéro ou la position d'une diode LED à partir de la matrice, il suffit d'écrire ceci :

```
numLed = X + (Xmax * Y)
```

Nous supposons que les deux valeurs X et Y sont les coordonnées de la diode LED désirée et que `Xmax` contient le nombre de diodes dans une ligne. Par mesure de précaution, nous conseillons de toujours faire un test avant la conversion pour vous assurer que les coordonnées fournies restent dans les limites de la matrice réelle.

Les matrices qui suivent la convention en « serpent » sont plus faciles à câbler, mais moins simples à gérer pour l'adressage. En effet, la conversion que nous venons de voir ne fonctionne que sur les lignes impaires. Pour les lignes paires, il faut utiliser la formule suivante :

```
numLed = (Xmax - X - 1) + (Y * Xmax)
```

La routine de conversion doit donc d'abord tester si la coordonnée en Y correspond à une ligne paire ou impaire afin de choisir la bonne formule. Il suffit de tester le bit le moins significatif, celui le plus à droite dans la valeur de la coordonnée Y.

Parmi les primitives indispensables pour afficher quelque chose dans une matrice, il y a celle de tracé de ligne. Elle reçoit en entrée les coordonnées d'un point de départ et celle d'un point d'arrivée, ce qui permet ensuite de

tirer un trait entre les deux, en allumant les diodes LED correspondantes. Sur une petite matrice telle que celle de notre exemple, seuls les traits verticaux, horizontaux et en diagonale auront un aspect satisfaisant. Tous les autres angles seront crénelés. Une astuce consiste à gérer la différence entre le point de départ et celui d'arrivée en augmentant de un la valeur de l'axe qui montre la plus grande différence, alors que l'autre axe reçoit une valeur d'augmentation moins importante. Nous pouvons ensuite commencer à allumer les diodes en cherchant à chaque fois la diode suivante par ajout des incréments dans les deux sens. Un des incréments vaudra une unité, alors que l'autre aura une valeur inférieure à un. Lorsque cette valeur accumulée viendra rejoindre une valeur entière, nous changerons la coordonnée selon cet axe.

Nous avons défini deux autres primitives pour tracer un carré vide et un carré plein. Vous remarquez que le code du carré vide est plus long que l'autre. Nous aurions pu utiliser la primitive de tracé de ligne pour réaliser ces deux figures, mais il est plus rapide de tracer des lignes dans une boucle, car cela évite de devoir faire des divisions.

Dans notre exemple, nous faisons afficher trois motifs différents qui se succèdent après un délai. Étudions le code source du Listing 17.7.

Comme les deux précédents, cet exemple utilise la version française de la définition de la classe, donc le fichier *apa102bangfr.py*.

Listing 17.7 : Code source d'une broche avec matrice de LED.

```
#!/usr/bin/env python3
# Broche avec matrice AdaFruit 8X8
# VF par O.E.

import time, random
from apa102bangfr import Apa102bangfr

broData = 23
broClock = 24
lumi = 1
lenX = 8
lenY = 8
nbrLED = lenX * lenY
demi = int(nbrLED / 2)
```

```
dureeMotif = 8.0
matri =
Apa102bangfr(nbrLED,broData,broClock,lumi)

def main():
   print("Matrice APA102 8x8 - Ctrl-C pour
quitter")
   tpsProchain = time.time() + dureeMotif
   motif = 1
   while True:
      if motif == 1:
         motiver1()
      if motif == 2:
         motiver2()
      if motif == 3:
         motiver3()
      if time.time() > tpsProchain:
         tpsProchain = time.time() + dureeMotif
         motif +=1
          if motif > 3:
            motif = 1

def motiver1():
     for cote in range(1,lenX):
       for i in range(0,lenX):
          carrer(i,i,cote,aleaCoul())
           matri.afficher()
          time.sleep(0.1)
       time.sleep(0.6)
       matri.changerTout((0,0,0))
     matri.changerTout((0,0,0))
     time.sleep(0.6)

def motiver2():
   start = 0
   for cote in range(lenX-1,0,-2):
```

```
      carrer(start,start,cote,aleaCoul())
      matri.afficher()
      time.sleep(0.08)
      start += 1
   #time.sleep(0.6)
   #matri.changerTout((0,0,0))

def motiver3():
   col = aleaCoul()
   for t in range(1,8):
      ligner(0,0,t,0,col)
      retenir()
   for t in range (1,8):
      ligner(0,0,8,t,col)
     retenir()
   for t in range (8,0,-1):
      ligner(0,0,t,8,col)
      retenir()
   for t in range(7,0,-1):
      ligner(0,0,0,t,col)
      retenir()

def retenir():
   matri.afficher()
   time.sleep(0.08)
   matri.changerTout((0,0,0))

def matricer(x,y,col):
   pixel = x + y*8
   if pixel < nbrLED and x < lenX:
      matri.changerLed(pixel,col)

def kRemplir(x,y,cote,col):
   for xp in range(x,x+cote):
      for yp in range(y,y+cote):
         matricer(xp,yp,col)
```

```
def carrer(x,y,cote,col):
   for xp in range(x,x+cote):
      matricer(xp,y,col)
   for xp in range(x,x+cote+1):
      matricer(xp,y+cote,col)
   for yp in range(y,y+cote):
      matricer(x,yp,col)
   for yp in range(y,y+cote):
      matricer(x+cote,yp,col)

def ligner(xs,ys,xe,ye,col):
   xl = xe - xs
   yl = ye - ys
   if xl > yl:
      xinc = 1.0
      yinc = yl/xl
    else:
      yinc = 1.0
      xinc = xl/yl
   x= float(xs)  ;  y = float(ys)
   while x  != xe or y  != ye:
      matricer(int(x),int(y),col)
      x += xinc
      y += yinc

def aleaCoul():
  r = 0;  g =0;  b=0
   while r+g+b == 0:
      r = random.randint(0,2) * 64
      g = random.randint(0,2) * 64
      b = random.randint(0,2) * 64
   return (r,g,b)

if __name__ == '__main__':
  try:
```

```
    main()
  except KeyboardInterrupt:
    pass
# tout eteindre
matri.changerTout((0,0,0))
matri.afficher()
```

La variable nommée `motif` détermine ce qu'il faut afficher. Elle change de valeur une fois que le temps écoulé correspond au contenu de la variable `dureeMotif`. Cette variable n'est testée qu'à la fin des trois animations, ce qui garantit que même les animations longues seront toujours vues au moins une fois en entier.

Les trois motifs exploitent les différentes primitives d'affichage en utilisant une couleur d'affichage choisie par la fonction **`aleaCoul()`** qui génère un mélange de couleurs pour chaque couleur primaire, mais exclut le noir. Le premier motif affiche une série de carrés dont les coins inférieurs sont calés sur la diagonale de la matrice. Au départ, les carrés sont petits puis ils grossissent à chaque remplissage de la diagonale jusqu'à occuper le carré de huit sur huit.

Le deuxième motif affiche une série de carrés imbriqués dans des couleurs variées. Le dernier motif affiche une ligne droite en balayage depuis un angle inférieur, dans le sens inverse des aiguilles d'une montre.

Nous avons dit que cette matrice pouvait servir de broche d'ornement. C'est réalisable en la pilotant avec une minicarte Pi Zero alimentée par piles. Vous pouvez aussi vous contenter de la poser sur votre bureau. Vous personnalisez les durées d'affichage en jouant sur les temps de pause. Vous aurez peut-être remarqué que la fonction d'affichage du carré plein n'était pas utilisée. Vous pouvez l'ajouter pour enrichir vos motifs et créer d'autres motifs pour encore plus de variété de l'animation.

Chapitre 18

L'identification par radiofréquences

DANS CE CHAPITRE :

- **Principe des communications RFID**
- **Lecture d'une carte RFID**
- **Un juke-box à cartes**

La technologie RFID, qui signifie *Radio Frequency IDentification* est utilisée partout, pour les badges d'accès aux bâtiments, pour les cartes prépayées des cafétérias, et même pour protéger contre les contrefaçons les billets de concerts et d'événements sportifs. Vous en trouverez aussi sous forme de bracelets d'accès aux parcs d'attractions, et même pour les cartes de donneurs de sang. Si votre animal domestique est « pucé », ce que vous lui avez fait installer est une petite capsule juste sous la peau du cou, contenant une étiquette RFID.

La polyvalence de la technologie RFID est telle que vous pourriez croire qu'il s'agit d'un équipement complexe, ce qui est le cas. Pourtant, à partir du moment où toute la complexité est prise en charge par un circuit dédié avec des fonctions prédéfinies par une classe pilote, l'utilisation dans un projet devient vraiment simple. Les cartes RFID permettent toutes sortes d'applications plus ou moins sérieuses, ce que nous allons voir dans ce chapitre.

Principe du RFID

Il existe trois grands types de technologies RFID, qui se distinguent d'abord par la plage de fréquences utilisée. Dans les trois cas, il y a deux partenaires : un badge ou une étiquette, et un lecteur. Le lecteur récupère

des bits de données depuis le badge, en les recevant par ondes radio. Aucune connexion n'est à réaliser entre lecteur et badge. Ce sont des badges passifs, c'est-à-dire qu'ils n'ont besoin d'aucune source d'énergie.

Certains modèles sont actifs, et contiennent donc une toute petite pile de maintien en veille. Ces modèles actifs sont utilisés lorsque l'on a besoin d'augmenter le rayon d'action, et donc la distance entre badges et lecteur.

C'est le lecteur qui émet un signal radio. Le badge, s'il est assez proche, est influencé par l'onde reçue. Il la transforme en une très faible quantité d'énergie, suffisante pour alimenter un microcircuit pendant quelques instants. Ce circuit transmet alors en réponse un certain nombre d'impulsions qui sont décodées par le lecteur pour obtenir une valeur numérique. La méthode de renvoi des données dépend de la technologie, mais le grand principe reste le même, hormis la fréquence.

Voici les trois technologies RFID disponibles, avec la plage de fréquences de chacune :

- **125-135 kHz.** Cette norme est celle utilisée par les vétérinaires et de nombreux systèmes de contrôle d'accès. La capacité de stockage est très limitée, puisqu'elle ne permet en général que de stocker une valeur sur 64 ou 128 bits. Les badges et étiquettes les plus répandus sont produits selon le standard EM4100/4200. La grande majorité des badges sont en lecture seule. Les balises HITAG notamment permettent cependant d'écrire des données avec un programmeur spécial.
- **13,56 MHz.** Cette technologie est celle des cartes intelligentes dites Smart Cards qui peuvent mémoriser un numéro de série, mais également d'écrire ou de lire un certain nombre de données. Les cartes sont dotées d'un système de sécurité constitué d'une clé de cryptage, ce qui protège les données de tout accès indésirable. Il existe de nombreux formats de cartes selon cette norme,

mais le plus répandu correspond à la norme MIFARE classique.

- **UHF 860-960 MHz.** Assez différente des deux autres technologies RFID, cette norme permet de lire plusieurs badges à la fois avec le même lecteur et porte la distance de lecture à environ 10 m. Elle équipe d'abord les systèmes d'automatisation des inventaires dans les entrepôts. Tous les objets d'une même palette peuvent être détectés et inventoriés en même temps. Les lecteurs sont dotés d'un émetteur à très forte puissance, ce qui oblige à prendre des précautions au niveau de la santé du personnel. La durée de travail avec un tel appareil est strictement limitée pour éviter une trop grande exposition aux rayonnements. Il existe même des systèmes à encore plus haute fréquence dans cette catégorie.

Dans ce chapitre, nous utilisons les cartes classiques MIFARE, qui correspondent au protocole ISO/IEC1443 A/MIFARE. Le lecteur est très bon marché, et les badges se présentent en général sous forme de cartes plastiques ou de porte-clés. Le format carte est le plus pratique, mais les autre conviennent aussi. La Figure 18.1 donne le schéma de fonctionnement d'un système RFID.

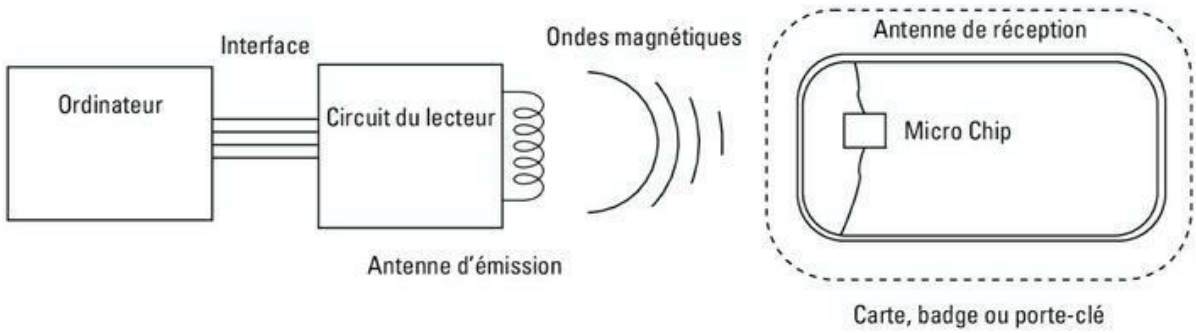


Figure 18.1 : Diagramme fonctionnel d'un système RFID.

L’antenne réceptrice est constituée d’une feuille métallique ou d’un fil très fin disposé en motifs concentriques formant une piste. Au niveau du lecteur, nous choisissons l’un des moins chers, le RFID-RC522. Vous en trouverez pour moins de 10 € chez tous les distributeurs. Ce modèle utilise le circuit MFRC522 de la société NXP (anciennement Philips). Le circuit est capable de dialoguer de plusieurs manières avec un ordinateur, mais la façon dont il est utilisé sur le lecteur oblige à utiliser le protocole SPI. Nous avons vu dans le précédent chapitre comment émuler ce protocole en envoyant des bits en série. Dans ce chapitre, nous choisissons d’utiliser l’interface SPI matérielle dont dispose le Raspberry Pi. Cela oblige à utiliser certaines broches dédiées du connecteur GPIO.

Le lecteur RFID que nous préconisons est proposé avec deux types de connecteurs mâles. Choisissez le connecteur dont les broches sortent à angle droit, car cela vous permet d’enficher le lecteur verticalement sur la plaque d’expérimentation, comme le montre la Figure 18.2.

Le montage vertical est fortement conseillé, car cela éloigne l’antenne réceptrice le plus possible des petites barrettes métalliques qui constituent les contacts de la plaque d’expérimentation. Tout métal a un effet sur la fréquence de résonance de l’antenne réceptrice, et donc sur la portée de l’identification. La Figure 18.3 propose le schéma de principe et le plan de câblage du lecteur sur le connecteur GPIO.

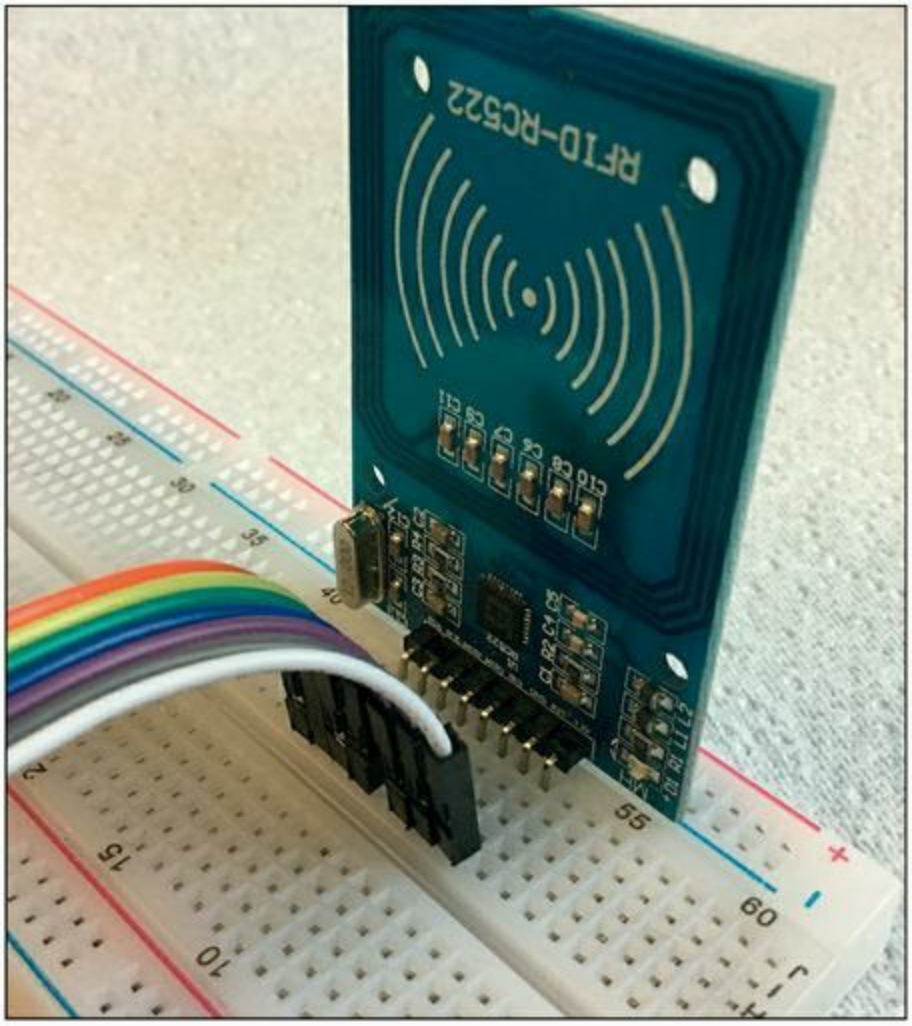


Figure 18.2 : Montage vertical du lecteur sur une plaque.

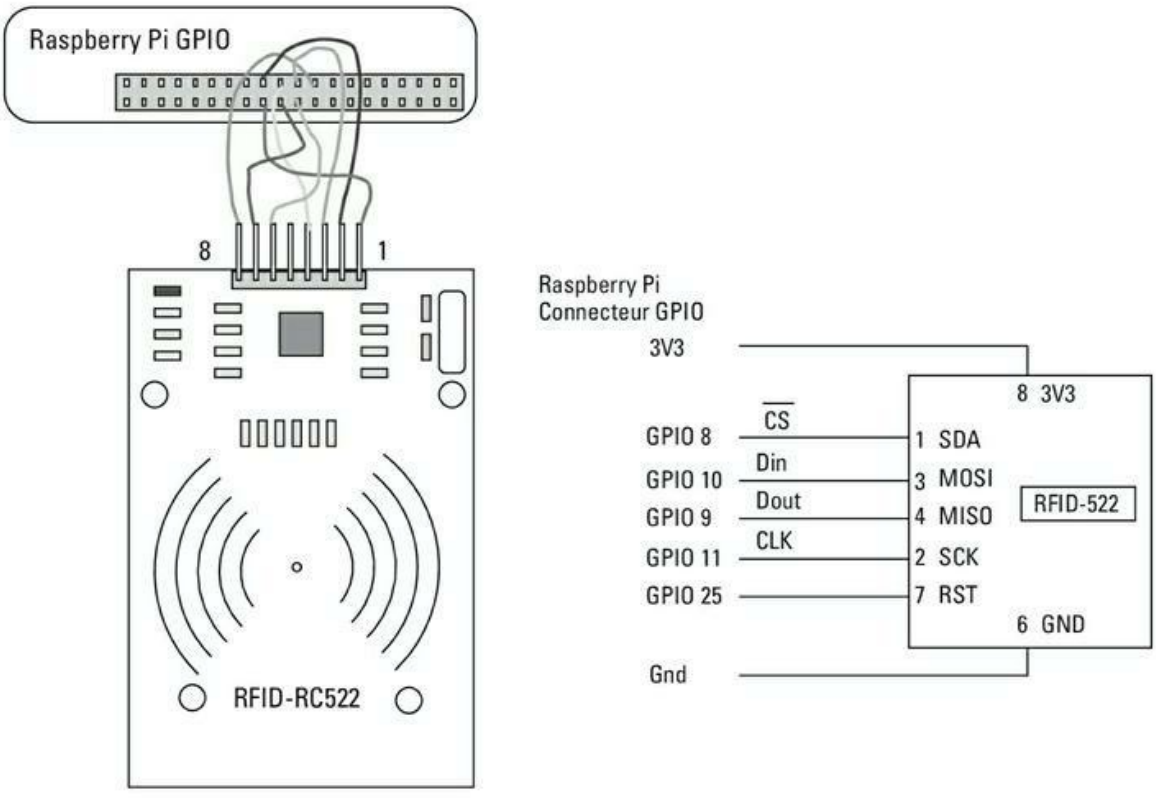


Figure 18.3 : Plan de câblage et schéma de principe du lecteur RFID.

N.d.T. : Les lecteurs n'ont pas tous les mêmes légendes de broches. La broche 3,3 V de notre schéma est parfois marquée VCC et la broche SDA est marquée NSS. La seule broche à ne jamais connecter est IRQ. Voici sur quelles broches du GPIO nous avons branché le modèle de Joy-It :

Signal Lecteur	Broche GPIO
Vcc ou 3,3 V	1 ou 17
RST	22
GND	6, 14 ou 20
MOSI	19
MISO	21
SCK	23
SDA/NSS	24

Une fois le lecteur relié à la carte RasPi, nous pouvons passer à la programmation. Mais il nous faut d'abord installer le sous-système python-dev avec la commande suivante dans une fenêtre de terminal :

```
sudo apt-get install python-dev
```

Évidemment, votre carte RasPi doit être connectée à Internet. Il est fort probable que votre système possède déjà la plus récente version, mais il est toujours préférable de vérifier.

Vous devez ensuite installer une librairie pour le protocole SPI, qui porte le nom *SPI-Py* et permet d'utiliser en langage Python l'interface matérielle SPI. Voici les quatre commandes à émettre dans la fenêtre de terminal :

```
git clone https://github.com/lthiery/SPI-Py.git
cd SPI-Py
sudo python setup.py install
sudo python3 setup.py install
```

Vous avez ainsi installé le code permettant d'exploiter le protocole SPI en Python 2 et en Python 3. La dernière opération consiste à activer l'interface SPI. Depuis le bureau, accédez aux préférences puis à l'application de configuration Raspberry Pi. Ouvrez la page **Interfaces** et activez l'option **SPI**. Vous devez ensuite redémarrer votre système.

Structure mémoire d'une carte MIFARE

Le standard MIFARE prévoit le stockage de 64 blocs de données, de 16 octets chacun. Certains blocs sont destinés aux données d'utilisation et d'autres contiennent des clés d'authentification, des numéros UID ou un identifiant de fabricant. Le numéro ID (*Unique IDentifier*) fait 4 octets de long. Malgré son nom, il ne peut donc pas être unique (il y a plus de 4 milliards de cartes en circulation), mais vous risquez plus souvent de gagner le gros lot de la loterie que de rencontrer physiquement le même UID que le vôtre. En pratique, vous pouvez donc le considérer unique.

Voici à quoi ressemblent les 12 premiers secteurs d'une carte MIFARE :

```
Sector 0 [203, 58, 164, 213, 128, 8, 4, 0, 98,
99, 100, 101, 102, 103, ↲
104, 105]
Sector 1 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0]
Sector 2 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0]
Sector 3 [0, 0, 0, 0, 0, 0, 255, 7, 128, 105,
255, 255, 255, 255, 255, 255]
Sector 4 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0]
Sector 5 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0]
Sector 6 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0]
Sector 7 [0, 0, 0, 0, 0, 0, 255, 7, 128, 105,
255, 255, 255, 255, 255, 255]
Sector 8 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0]
```

```
Sector 9 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0]
Sector 10 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0]
Sector 11 [0, 0, 0, 0, 0, 0, 255, 7, 128, 105,
255, 255, 255, 255, 255, 255]
```

Le premier secteur est non modifiable. Il contient les 4 octets de l'UID. Les secteurs qui ne contiennent que des zéros sont ceux dans lesquels vous pouvez stocker des données. Chaque groupe de quatre secteurs utiles est précédé d'un secteur qui contient de clés pour l'autorisation d'accès en lecture et en écriture à ces quatre secteurs suivants. Vous n'écrivez jamais directement dans les secteurs de contrôle, mais le fait d'écrire dans les autres modifie les valeurs. Au départ, comme indiqué dans l'exemple, ce sont les valeurs des clés par défaut qui sont présentes. Lorsque vous prévoyez d'écrire sur un support RFID, n'oubliez jamais que vous ne pouvez écrire que dans les secteurs qui ne contiennent pas de clés.

Les secteurs-clés vers lesquels vous ne pouvez pas écrire sont les suivants :

```
0, 3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47,
51, 55, 59, 63
```

Il reste encore à installer une copie du fichier qui définit la classe nommée **`Rc522rfid`** dans le dossier dans lequel vous allez créer vos projets. Vous trouverez le fichier dans l'archive des exemples du livre. (Revoyez l'introduction qui explique comment télécharger l'archive depuis le site de l'éditeur.) Nous n'avons pas jugé utile de présenter le code source de cette classe ici, car il est très long et assez complexe. Sachez que le code vous interdit, comme la norme le suggère, d'écrire dans les secteurs clés.

Le dialogue avec le lecteur par l'interface SPI n'est pas des plus simples, mais il n'est pas nécessaire de le maîtriser pour savoir exploiter ce lecteur. Ce dialogue consiste en la préparation de valeurs numériques qui sont stockées dans des registres internes ou dans des zones mémoire, dans un ordre précis (comme indiqué dans la fiche technique du circuit). Pour vous simplifier la vie, nous avons ajouté à la fin du fichier de la classe des définitions de méthodes qui suffisent à exploiter le lecteur en lecture et en écriture. Voici donc les cinq méthodes dont vous avez besoin de connaître l'utilisation :

- **RC522_WaitForCardRemoved**. Cette méthode place le programme en pause jusqu'à ce que la

carte ait été éloignée du lecteur. Notez que la lecture est assez rapide. Le problème est que si vous maintenez longtemps la carte ou le badge devant le lecteur, les appels suivants à la fonction de détection renvoient la valeur False, comme s'il n'y avait pas de carte. C'est le résultat du mode de fonctionnement du circuit de contrôle. Pour contourner ce problème, nous avons prévu cette méthode qui attend que deux essais successifs de lecture renvoient une apparente absence de carte. Il s'agit d'un code bloquant, c'est-à-dire que rien d'autre ne peut s'exécuter tant que la carte n'est pas éloignée.

» **RC522_ReadCard**. Procède à la lecture et au renvoi de l'identifiant de la carte. En cas d'erreur de lecture ou d'absence de carte, la méthode renvoie la valeur – 1.

» **RC522_GetCard**. Attend qu'il n'y ait plus de carte dans le lecteur puis attend qu'une nouvelle carte soit détectée. Dans ce cas, elle lit l'identifiant et le renvoie. Il s'agit d'une méthode bloquante.

» **RC522_GetSector**. Lit un bloc de données de 16 octets et renvoie cette valeur. Vous devez fournir la clé d'autorisation et le numéro du secteur à lire. Cette méthode est bloquante. Elle attend qu'une carte valable soit détectée.

» **RC522_WriteSector**. Écrit un bloc de données de 16 octets dans un secteur. Vous fournissez la clé et le numéro de secteur à écrire. Cette méthode est bloquante et attend qu'une carte

> valable soit détectée. Elle interdit l'écriture dans les secteurs clés.

Voyons maintenant comment mettre cela en pratique : le Listing 18.1 montre comment simplement lire l'identifiant UID d'une carte. Il vous sera utile pour noter les identifiants des cartes que vous allez utiliser dans la suite.

Listing 18.1 : Programme de lecture de l'identifiant d'une carte RFID.

```
#!/usr/bin/env python
# Token reader by Mike Cook
# VF O.E.

from rc522rfid import Rc522rfid

lectRF = Rc522rfid()

print ("Lecteur de code UID de carte RFID")
print ("Ctrl-C pour quitter.")
while 1: # Boucle infinie
   UIDcarte = lectRF.RC522_GetCard()
   print(hex(UIDcarte))
```

L'identifiant UID est une valeur sur 16 bits que nous affichons dans le format hexadécimal. En effet, l'affichage décimal n'aurait aucun intérêt parce que vous verriez un mélange de valeurs positives et négatives, selon l'état du bit le plus significatif. Nous savons donc lire l'identifiant, mais ne pourrait-on pas se servir de cette valeur pour déclencher une action, par exemple pour lancer la lecture d'un morceau de musique ?

Avant de passer à la suite, faites lire les UID de la poignée de cartes ou de porte-clés dont vous disposez en notant l'identifiant de chacune.

Un premier juke-box RFID

Voyons comment utiliser plusieurs cartes RFID pour déclencher la lecture d'un morceau de musique parmi plusieurs. Nous allons supposer que les fichiers sont stockés dans le sous-répertoire nommé *Music* de votre répertoire personnel. Dans votre répertoire personnel */home/pi/*, créez un sous-répertoire nommé *Music* (pour ne pas perturber le contenu actuel de votre répertoire *Musique*).

Pour les essais, vous pouvez récupérer des boucles depuis le sous-répertoire de Scratch suivant :

```
/usr/share/scratch/Media/Sounds/Music Loops
```

Vous pouvez également y placer quelques fichiers MP3 de votre discothèque. Si ce n'est pas encore fait, faites identifier chacune de vos cartes par le programme précédent et notez l'identifiant sur chaque carte ou sur une feuille de papier avec une marque sur chaque carte pour faire le lien. Vous pouvez ensuite saisir ou charger le programme du Listing 18.2.

Listing 18.2 : Un juke-box RFID.

```
#!/usr/bin/env python
# Simple RFID Jukebox by Mike Cook
# VF par O.E.

from rc522rfid import Rc522rfid
import pygame

pygame.mixer.quit()
pygame.mixer.init(frequency=22050, size=-16,
channels=2, buffer=512)
lectRF = Rc522rfid()

# Indiquez ici les UID hexa de vos cartes:
cartes =
[0xf6690ebb,0xcb3aa4d5,0x9c35fddd,0x8cd32dde,0xc68

ficSon = [
"Drum","DrumMachine","DrumSet1","DrumSet2","Eggs"]
```

```
def main():
  print ("Lecteur Jukebox RFID simple")
  print ("Ctrl-C pour quitter.")
  while 1:                                # toujours
    IDcarte = lectRF.RC522_GetCard()
    pygame.mixer.music.stop()
    for i in range(0,len(cartes)):
      if IDcarte == cartes[i]:
          try  :

pygame.mixer.music.load("/home/pi/Music/"+str(ficS

          pygame.mixer.music.play()
          print("Lecture de:-
"+str(ficSon[i])+".mp3")
        except  :
          print("Erreur:-
/home/pi/Music/"+str(ficSon[i])+".mp3")

#
if __name__ == '__main__':
  try:
    main()
  except KeyboardInterrupt:
    pass
```

Comme pour tous les autres exemples, si vous avez un souci et avez vérifié que la saisie ne comporte aucune erreur, utilisez le fichier fourni dans l'archive à télécharger sur le site. Il est toujours possible qu'une retouche ait été apportée entre l'envoi du livre chez l'imprimeur et le bouclage des exemples.

Dans le code source, descendez jusqu'à la ligne qui définit la liste `cartes` et remplacez les valeurs hexa par celles que vous avez notées, sans oublier le préfixe 0x devant chaque valeur. Modifiez également la liste suivante, `ficSon`, pour qu'elle indique les noms des fichiers que vous voulez faire jouer (sans l'extension .mp3). Si vous démarrez le programme, il suffit d'approcher une carte pour provoquer la lecture du

morceau correspondant dans l'ordre des deux listes. Autrement dit, la première carte déclarée va provoquer la lecture du premier morceau de la liste.

C'est un bon début, mais cela signifie qu'il faut modifier le code source chaque fois que vous voulez écouter d'autres morceaux. Il serait plus intéressant de faire mémoriser le nom des morceaux à lire directement dans les cartes.

Un juke-box RFID plus versatile

Vous pouvez stocker une information sur la carte RFID, mais il faut écrire un programme pour réaliser cette écriture. Cela revient à déclarer la carte dans le projet. Dans l'exemple précédent, la déclaration était figée dans le code source, en indiquant l'identifiant de la carte dans une liste. Les noms des fichiers audio étaient aussi figés dans une autre liste et stockés dans un endroit particulier. Dans cette version améliorée, ces éléments vont être définis dans le programme de déclaration.

Nous allons donc disposer de deux programmes : un pour lire la musique mentionnée dans la carte présentée et un autre pour stocker un nom de fichier dans le segment de données de la carte. Le programme de déclaration que nous allons découvrir commence par lire les noms de tous les fichiers qu'il trouve dans le sous-répertoire Music puis il les présente l'un après l'autre, afin de vous permettre d'en écarter certains. Si vous sélectionnez un nom de fichier, il est stocké avec son extension dans les deux segments 4 et 5 de la carte. La conséquence et que les noms de fichiers ne doivent pas avoir une longueur supérieure à 32 octets, extension .mp3 comprise (même si cette dernière n'est pas visible dans votre configuration). Découvrons donc maintenant le programme de déclaration (Listing 18.3).

Listing 18.3 : Programme de déclaration pour le juke-box RFID.

```
#!/usr/bin/env python
# RFID Jukebox Enroll - stocke noms dans secteurs
4&5
# par Mike Cook
# VF par O.E.

from rc522rfid import Rc522rfid
import os
```

```
lectRF = Rc522rfid()

def main():
   print ("Declareur pour Jukebox RFID")
   print ("Ctrl-C pour quitter.")
  listerSons()
   for numFic in range(0,len(listeSons)):
      action = ""
      print( listeSons[numFic] )
      print("Touche (e) ou Entree pour ignorer")
      action = input()
      if action == "e" :
         declarer(listeSons[numFic])

def declarer(nomFic):
   print("Stockage de", nomFic, ". Presentez une
carte.")
   data1 = []
   data2 = []
   for i in range(0,16): # Remplir de zeros
      data1.append(0)
      data2.append(0)
   for i in range(0,len(nomFic)):
      if i < 16:
         data1[i] = ord(nomFic[i])
       else:
         data2[i-16] = ord(nomFic[i])
   # Cle par defaut pour authentification
   key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]
   lectRF.RC522_WriteSector(key,4,data1)
   lectRF.RC522_WriteSector(key,5,data2)
   print("fin de declaration\n")

def listerSons():
   global listeSons
```

```
    chemin = os.path.abspath("/home/pi/Music/")
    # Obtention liste de noms
    listeSons = [fn for fn in
next(os.walk(chemin))[2]]
    list.sort(listeSons)         # Tri alpha
    print (len(listeSons),"fichiers.")

#
if __name__ == '__main__':
  try:
    main()
  except KeyboardInterrupt:
    pass
```

La fonction nommée **`listerSons()`**, à la fin du code source, remplit une liste avec les noms des fichiers trouvés dans le sous-répertoire désigné par le chemin d'accès stocké dans la variable `chemin`, en utilisant la méthode de librairie standard **`os.walk()`**. La liste est triée dans l'ordre alphabétique puis le nombre de fichiers trouvés est indiqué. De retour dans la fonction principale **`main()`**, nous affichons les noms de fichiers l'un après l'autre en vous proposant de frapper la touche **e** (minuscule) suivie de la touche Entrée pour envoyer le nom sur la carte.

Pour ignorer le fichier, vous frappez directement la touche Entrée afin de passer au suivant. Pour chaque fichier sélectionné, nous appelons la fonction **`declarer()`** qui découpe le nom en deux parties de 16 octets puis écrit les deux blocs dans les secteurs choisis sur la carte.

Une fois qu'une carte est déclarée, nous conseillons de la marquer pour savoir ce qu'elle contient. Vous pouvez ajouter une étiquette ou coller une miniature de la pochette du disque correspondant. Vous pouvez ainsi vous faire un véritable juke-box piloté par cartes.

Une fois que les cartes sont déclarées, il ne reste plus qu'à voir le programme de lecture (Listing 18.4).

Listing 18.4 : Programme de lecture de cartes RFID déclarées.

```
#!/usr/bin/python3
# RFID Jukebox2 par Mike Cook
# VF par O.E.
```

```
from rc522rfid import Rc522rfid
import pygame

pygame.mixer.quit()
pygame.mixer.init(frequency=22050, size=-16,
channels=2, buffer=512)
lectRF = Rc522rfid()

def main():
   print ("RFID Jukebox avec cartes declarees")
   print ("Ctrl-C pour quitter.")
   while 1:          # Infiniment
      aLire = lireCarte()
      pygame.mixer.music.stop()
       try  :

pygame.mixer.music.load("/home/pi/Music/"+aLire)
         pygame.mixer.music.play()
         print("Lecture de:- "+aLire)
       except  :
         print("Pas de fichier:-
/home/pi/Music/"+aLire)
      lectRF.RC522_WaitForCardRemoved()

def lireCarte():
   # Cle d'authentification par defaut
   key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]
   block1 = lectRF.RC522_GetSector(key,4)
   block2 = lectRF.RC522_GetSector(key,5)
   if block1 != -1 and block2 != -1:
      nomFicAjouer = rabouter(block1,block2)
      return nomFicAjouer
    else:
     return("Erreur de lecture de carte")
```

```
def rabouter(s1,s2):
    nom = ""
    i = 0
    pasFini = True
    while pasFini and i < 32:
        if i<16:
          c = s1[i]
        else:
          c = s2[i-16]
       i+=1
        if c  !=0:
         nom = nom + chr(c)
        else:
         pasFini = False
    return nom

#
if __name__ == '__main__':
  try:
    main()
  except KeyboardInterrupt:
    pygame.mixer.quit()
```

La fonction principale se place en attente de détection d'une carte puis elle lit le nom de fichier qu'elle y trouve et le stocke dans la variable `aLire`. Nous prenons la précaution d'interrompre toute émission sonore qui serait encore en cours (**`music.stop()`**) puis nous tentons de lancer la lecture. Si le fichier n'est pas trouvé, nous affichons un message et attendons que la carte ait été éloignée. La fonction **`lireCarte()`** utilise la clé par défaut pour lire les secteurs 4 et 5. La fonction **`rabouter()`** reconstruit le nom de fichier à partir du contenu des deux blocs de données.

Lorsque vous aurez terminé vos essais, vous pourrez installer le lecteur dans un petit boîtier en plastique ou en bois, en utilisant de préférence des vis nylon pour ne pas influencer l'antenne.

Pour aller plus loin

Vous pouvez facilement changer le nom du sous-répertoire dans lequel se trouvent les fichiers à lire. Un autre projet facile à réaliser à partir de ce juke-box est un ours en peluche qui raconte des histoires. Il suffit de placer le lecteur de carte dans son ventre et de prévoir une série de cartes. Un enfant peut alors choisir une carte chaque soir. Pour encore plus de réalisme, vous pouvez placer le haut-parleur dans la peluche.

Dans le sous-dossier US18 du dossier contenant ces exemples, vous trouverez plusieurs fichiers supplémentaires dont vous pouvez vous inspirer.

PARTIE 6
Les Dix Commandements

DANS CETTE PARTIE :

- Télécharger et installer dix superbes logiciels pour votre Raspberry Pi, parmi lesquels des jeux, des logiciels de création graphique et des outils de productivité.
- Puiser de l'inspiration auprès de dix projets innovants pour le Raspberry Pi, parmi lesquels un cardiogramme, un canon à T-shirts et un robot joueur d'échecs.
- Résoudre les problèmes les plus courants de votre Raspberry Pi, intervenir sur les paramètres avancés et connecter un périphérique de stockage externe depuis la ligne de commande Linux.

Chapitre 19

Dix logiciels sélectionnés pour le Raspberry Pi

DANS CE CHAPITRE :

- **Logiciels de jeux**
- **Logiciels éducatifs**
- **Mathématiques, fractales et jeux**

Un des points forts du système Linux qui anime le Raspberry Pi est qu'il vous offre littéralement l'accès à des milliers de logiciels (des paquetages) téléchargeables et installables dans l'instant même. Ce chapitre vous propose une très rapide sélection de dix logiciels pour vous donner une idée de la variété des domaines disponibles.

Avant d'installer quoi que ce soit, émettez la commande suivante pour vérifier que le cache local contenant la liste des logiciels est à jour :

```
sudo apt-get update
```

Le genre de logiciels que vous allez installer est aussi personnel que le type de musique que vous préférez. Servez-vous de nos suggestions comme d'un point de départ ; poursuivez la découverte par vous-même. La procédure de recherche et d'installation de logiciels a été présentée dans les Chapitres 4 et 5.

Penguins Puzzle

Penguins Puzzle (Figure 19.1) est un jeu de plateau 3D dans lequel vous devez guider un pingouin (ou un manchot ?) jusqu'à la sortie sans le laisser tomber dans l'eau glacée. Les flèches du curseur servent à déplacer

l'animal, la touche Z à zoomer (un peu) et la touche R à recommencer au début du niveau courant parmi les 50 à réussir. Vous sortez du jeu à tout moment par la touche Echap.

Figure 19.1 : *Penguins Puzzle* est un agréable labyrinthe en 3D.

Voici comment installer ce jeu :

```
sudo apt-get install penguinspuzzle
```

Et voici comment le démarrer depuis l'interpréteur :

```
penguinspuzzle
```

Ce logiciel est un chariciel (*charityware*) : vous êtes libre de faire un don à une œuvre de charité si vous appréciez le travail. Le site officiel du jeu est à l'adresse http://penguinspuzzle.appspot.com/.

FocusWriter

Que vous ayez décidé d'écrire le roman incontournable de la prochaine rentrée littéraire ou plus simplement que vous souhaitiez pouvoir rédiger

vos textes sans être distrait par des boutons et des menus partout, découvrez l'application FocusWriter. Ce traitement de texte vous invite à rester focalisé sur l'écriture. Dans le mode de travail normal, la seule chose affichée à l'écran, c'est votre prose.

Quand vous amenez le pointeur de souris en haut d'écran, vous faites apparaître les menus d'enregistrement et de paramétrage. Pour soutenir votre motivation, vous définissez dans les préférences un temps à consacrer ou un nombre de mots minimal à produire par jour. Les valeurs de suivi apparaissent en bas d'écran quand vous y amenez le pointeur.

Voici comment installer ce programme :

```
sudo apt-get install focuswriter
```

Vous démarrez FocusWriter depuis l'environnement graphique en le cherchant dans le sous-menu **Bureautique** du menu général des programmes.

Le site officiel est ici : http://gottcode.org/focuswriter/.

Mathematica

Mathematica est un logiciel de la famille CAS, ou *Computer Algebra System*. Il est préinstallé avec le système Raspbian. C'est un des meilleurs logiciels pour partir à l'exploration des nombres, des mathématiques, des graphismes à plusieurs dimensions et de la musique.

Pour le démarrer, ouvrez le menu général, le sous-menu **Programmation** et cliquez **Mathematica**. Après l'écran de bienvenue, vous voyez deux fenêtres apparaître : une fenêtre de saisie vide et une fenêtre vous proposant de visiter trois sites Web. Cliquez dans la fenêtre d'édition pour qu'elle passe au premier plan et saisissez par exemple la formule `2^8` puis validez par Entrée. Cette formule signifie « deux à la puissance huit ». Pourtant, vous n'obtenez pas la réponse après avoir validé. Pour demander au logiciel d'évaluer la formule et d'afficher la réponse, il faut utiliser la combinaison Maj + Entrée.

Mathematica est très puissant. Vous pouvez par exemple lui demander de développer une équation :

```
Expand[(1+x)^6]
1 + 6x + 15x2 + 20x3 + 15x4 + 6x5 + x6
```

Il sait également tracer des graphiques, comme cette courbe paramétrique :

```
For[n=1, n<4, n++,
ParametricPlot[ {Sin[n t], Sin[(n+1) t]}, {t, 0,
2Pi}] Print]
```

Vous devez être patient pour obtenir l'affichage de certains résultats graphiques complexes. En effet, Mathematica sait générer des graphiques en 3D :

```
SphericalPlot3D[Sin[t] Cos[t] Sin[f], {t, 0, Pi},
{f, 0, 2 Pi}]
```

Le résultat de cette équation est visible dans le bas de la Figure 19.2.

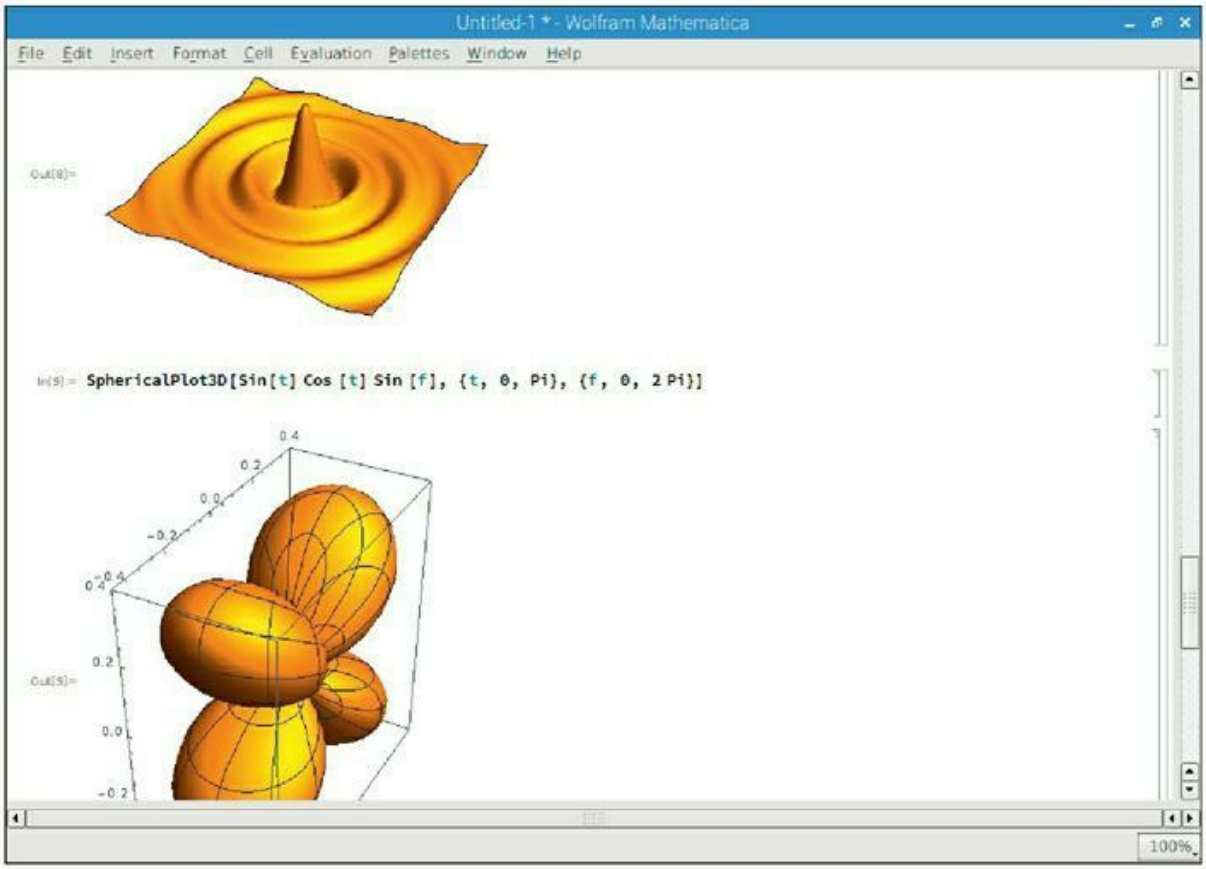

Untitled-1 * - Wolfram Mathematica
File Edit Insert Format Cell Evaluation Palettes Window Help
In[9]:= SphericalPlot3D[Sin[t] Cos [t] Sin [f], {t, 0, Pi}, {f, 0, 2 Pi}]
100%

Figure 19.2 : Deux exemples de graphiques créés par Mathematica.

Un des auteurs de ce livre adore le résultat de la formule suivante :

```
Plot3D[Sin[Sqrt[x^2 + y^2]]/Sqrt[x^2 + y^2],
{x, -6 Pi, 6 Pi}, {y, -6 Pi, 6 Pi},
Boxed -> False, Mesh -> False, PlotPoints -> 60,
```

```
PlotRange -> All, Axes -> False]
```

Nous avons prévu un petit fichier contenant cette formule pour vous éviter de la saisir. Récupérez-le dans les exemples et collez la formule dans Mathematica puis demandez son évaluation.

Invaders 3D

Vous êtes peut-être grand amateur des premiers jeux vidéo des années 1970 et 1980. Ce jeu a choisi d'exploiter des graphiques de type ligne comme le fameux *Asteroids* en revisitant l'autre grand classique qu'est *Space Invaders*. Le rendu 3D donne l'illusion que les aliens s'approchent graduellement. Vous utilisez les flèches du curseur pour les abattre et vous déplacer pour les esquiver. Le jeu est captivant et limpide. Vous faites feu avec la barre d'espace.

Voici comment installer ce programme :

```
sudo apt-get install xinv3d
```

Pour démarrer *Invaders 3D*, cherchez dans le sous-menu **Autres** du menu des programmes (ou tapez **xinv3d** sur la ligne de commande).

Fraqtive

Une fractale est un motif graphique dont les contours sont infiniment détaillés sous l'effet d'une formule mathématique. Lorsque vous vous approchez d'une figure de Mandelbrot (regardez à gauche sur la Figure 19.3), vous retombez sur le même motif, aussi loin que vous zoomiez. Fraqtive permet d'explorer l'univers des fractales et de générer des images que vous pouvez enregistrer pour vous en servir comme fond d'écran sur le Raspberry Pi (voir le Chapitre 4). Le logiciel est doté d'un tutoriel pour vous aider à faire vos premiers pas.

Voici comment installer ce programme :

```
sudo apt-get install fraqtive
```

Pensez à répondre à la demande de confirmation, car ce logiciel occupe environ 1 Mo d'espace sur la carte.

Pour démarrer Fraqtive, cherchez dans le sous-menu **Éducation** du menu des programmes (ou tapez **xinv3d** sur la ligne de commande).

Voyez aussi la page Web du logiciel :

http://fraqtive.mimec.org/

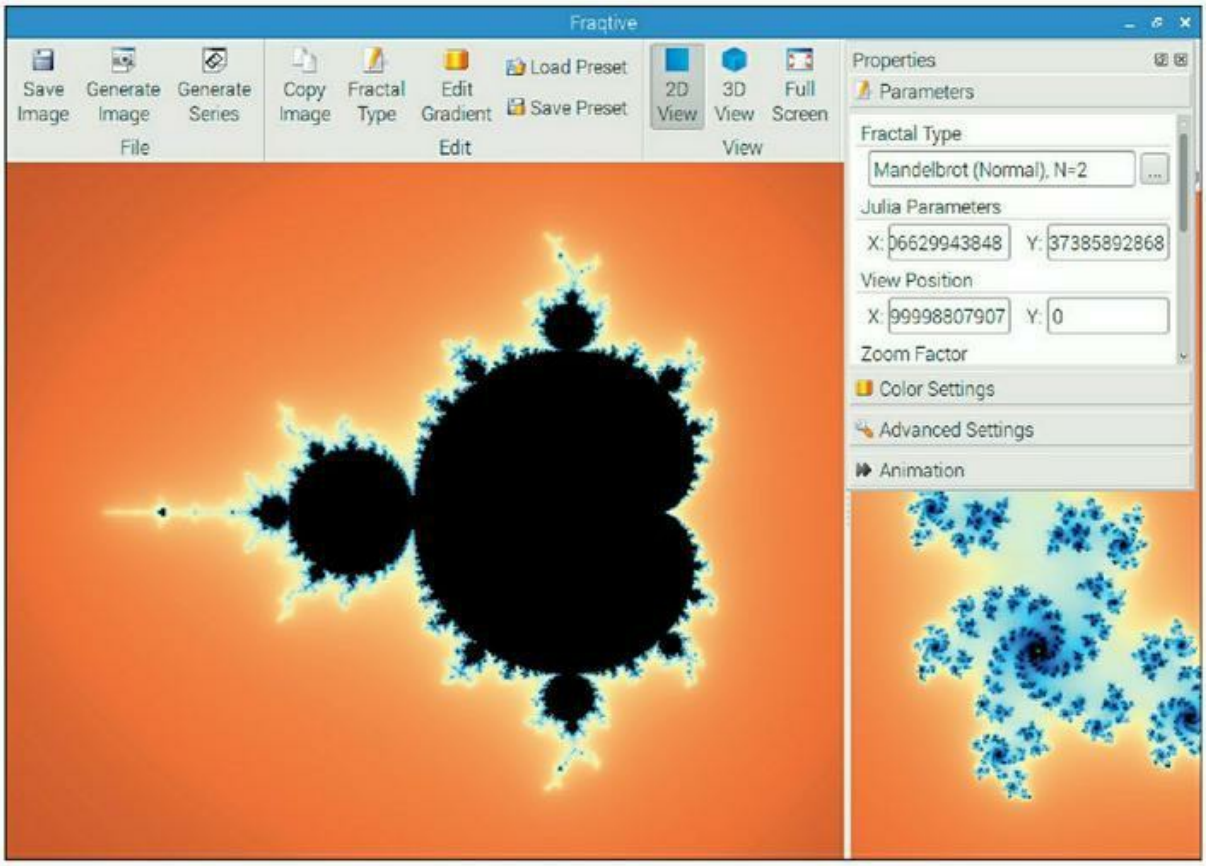


Figure 19.3 : Générez de superbes fractales colorées avec Fraqtive (Michal Mecinski).

Tux Paint

Tux Paint (Figure 19.4) est un logiciel de dessin bitmap dont l'interface a été simplifiée pour devenir accessible aux enfants. Il réunit des outils permettant de créer des dessins en quelques minutes sur le Raspberry Pi. Vous pouvez tracer à main levée et insérer des formes et des lignes comme tous les logiciels de dessin, mais Tux Paint possède un outil magique pour créer des effets (briques, fleurs, arcs-en-ciel, boules de neige, ondes et distorsions créatives). Le tampon sert à poser des images sur le canevas (animaux, pingouins, chapeaux, aliments et instruments de musique).

Le nom Tux Paint rappelle Tux, qui est le nom de la mascotte officielle du noyau Linux. Ce programme a été créé par plus de 300 contributeurs bénévoles du monde entier et a été téléchargé des dizaines de millions de fois.

Voici comment installer ce programme :

```
sudo apt-get install tuxpaint
```

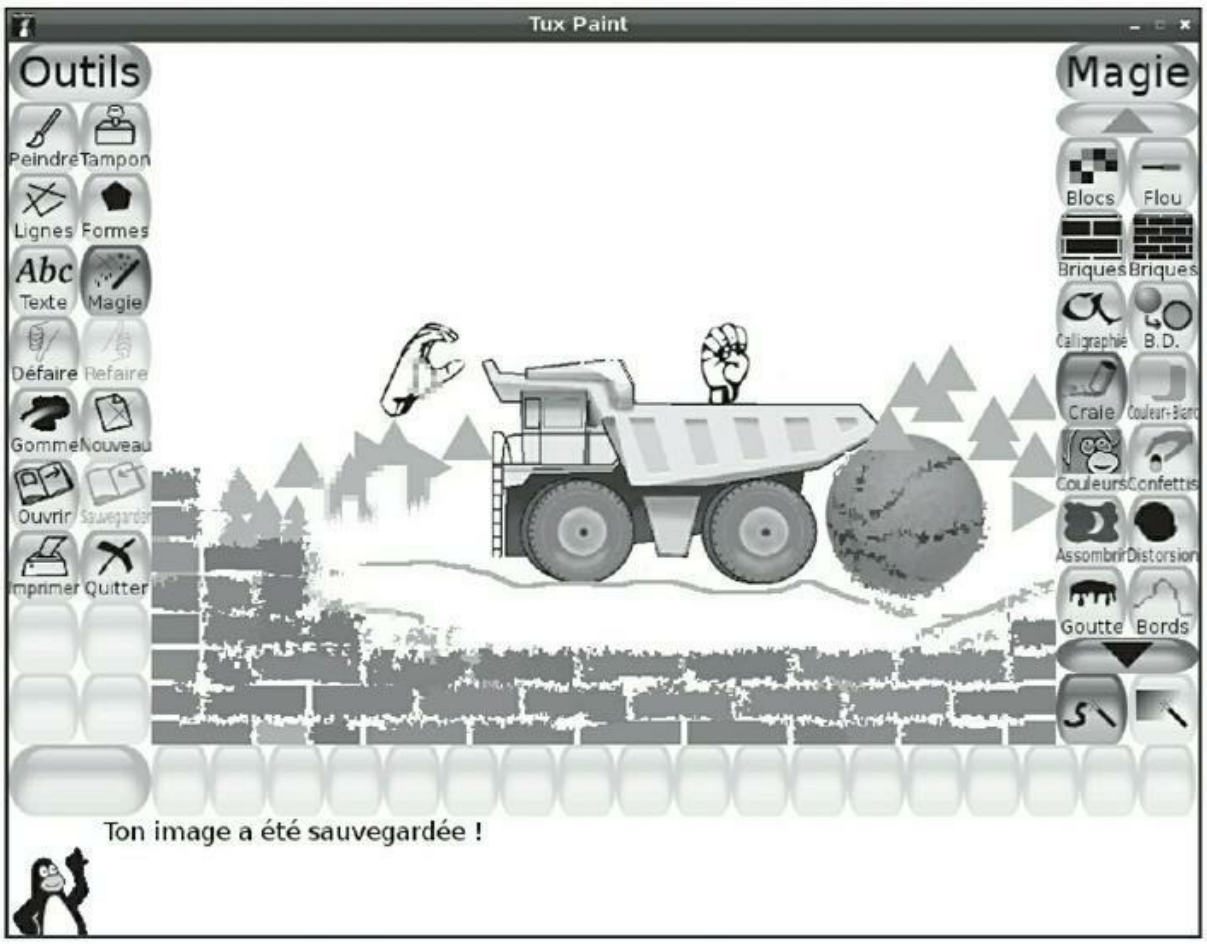


Figure 19.4 : Avec Tux Paint, chaque enfant devient un artiste.

Une fois installé, une mention permettant de démarrer le programme est disponible dans le sous-menu **Éducation** du menu général des programmes en mode graphique.

Voici l'adresse du site officiel : www.tuxpaint.org.

Grisbi

Vous pouvez gérer vos comptes personnels sur votre Raspberry Pi avec le logiciel gratuit Grisbi. Il sait gérer les versements périodiques comme les encaisses et décaisses sporadiques. Parmi les logiciels libres de cette catégorie, Grisbi nous a semblé le plus facile à configurer, à utiliser et à maintenir à jour. La plupart des banques vous permettent de télécharger vos écritures dans un format exploitable par Grisbi, ce qui peut vous éviter de fastidieuses séances de ressaisie.

Voici comment installer ce programme :

```
sudo apt-get install grisbi
```

Une fois installé, une mention permettant de démarrer le programme est disponible dans le sous-menu **Bureautique** du menu général des programmes.

Beneath a Steel Sky

Beneath a Steel Sky (sous un ciel de plomb) est un jeu de science-fiction (Figure 19.5) dans lequel Robert Foster avait survécu enfant au crash d'un hélicoptère puis avait été élevé par des aborigènes australiens dans une contrée désolée nommée *The Gap*. Devenu adulte, Robert est kidnappé par une meute de forces armées dans un autre hélicoptère et ramené de force en ville. Il réussit à échapper à ses ravisseurs, et c'est à ce moment que vous prenez le contrôle du personnage dans sa découverte de cette jungle urbaine. Pourquoi est-il là ? Qui pilote tout cela ?

La navigation se fait par pointer/cliquer : vous devez résoudre des énigmes en interagissant avec le décor au moyen du pointeur de souris pour cliquer sur les objets et les personnages. Le bouton gauche de souris sert à examiner les objets et le bouton droit à choisir une action (ouvrir une porte, ramasser un objet, regarder par une fenêtre, *etc*.). Cliquer sur un personnage permet de lui parler en choisissant parmi plusieurs phrases. Vous ouvrez votre inventaire en amenant le pointeur en haut de fenêtre et vous sortez de la scène en cours en cliquant sur l'issue de sortie.

Figure 19.5 : *Beneath a Steel Sky*, un jeu de science-fiction interactif.

La séquence d'ouverture du jeu et les dialogues pleins d'esprit vous invitent à plonger dans l'aventure. Si vous êtes bloqué, la solution des énigmes est disponible sur le Web.

Ce jeu a eu beaucoup de succès lors de sa sortie en 1994. Il a été transformé en graticiel en 2003. Voici comment l'installer sur votre Raspberry Pi :

```
sudo apt-get install beneath-a-steel-sky
```

Pour démarrer le programme, voyez dans le sous-menu **Jeux** du menu général des programmes.

Sense HAT Emulator

La carte d'extension Sense HAT est un produit officiel de Raspberry Pi. Au départ, cette carte était prévue pour faire des expériences pédagogiques dans la station spatiale internationale ISS. Comme toutes les cartes boucliers au format HAT, elle se branche directement sur le connecteur GPIO. Elle réunit une matrice de LED RGB de 8 sur 8, un joystick à cinq boutons et une série de capteurs de température, de pression et d'humidité ainsi qu'une boussole. Vous pouvez écrire des programmes pour cette carte en langage Python.

Vous pouvez vous faire une idée de ce que permet la carte grâce à l'émulateur visible dans la Figure 19.6. Il est livré avec le système Raspbian. Sur le bureau PIXEL, vous le trouverez dans le menu **Programmation**.

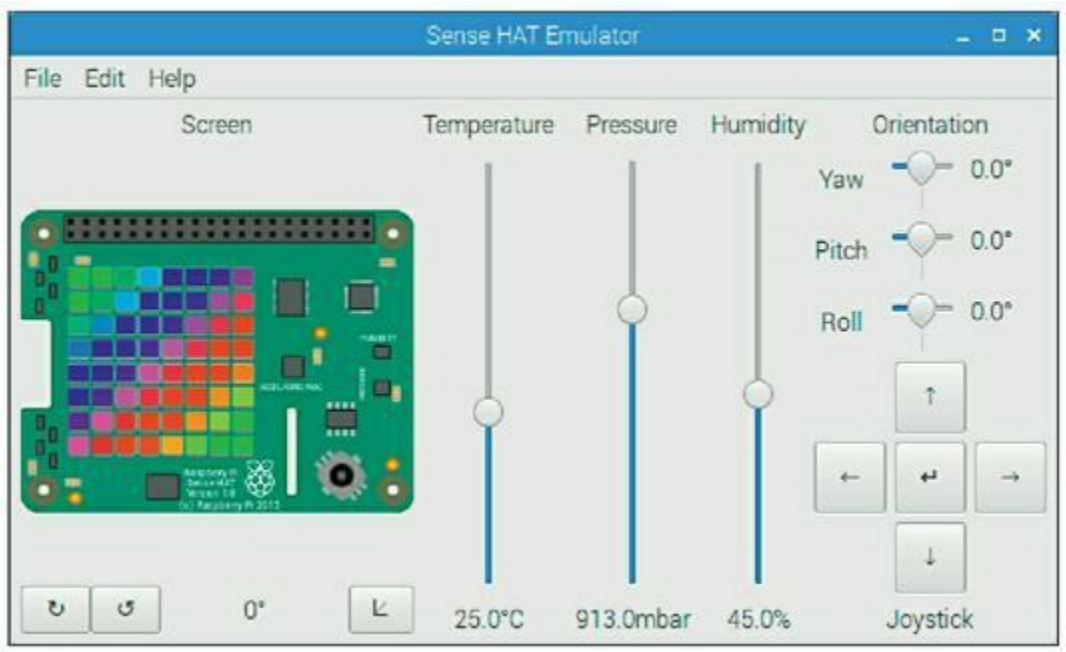


Figure 19.6 : L'émulateur de la carte Sense HAT.

Cet émulateur permet de tester les broches d'entrée et de sortie de la carte pour vérifier vos programmes. Il est livré avec un certain nombre d'exemples auxquels vous avez accès par le menu File/Fichier.

Une fois que vous avez créé un programme qui vous semble intéressant, vous pouvez passer à l'acquisition de la carte Sense HAT qui coûte dans les 40 €.

Brain Party

Si vous avez besoin de vous relaxer les neurones entre deux séances de programmation, le jeu Brain Party est la solution. Il s'agit d'une collection de minijeux pour solliciter votre matière grise. Vous devez réaliser cinq tests au hasard pour obtenir un score de « cerveau » puis vous pouvez jouer à chacun des minijeux déverrouillés grâce à votre test. Ces jeux vont mettre au défi votre mémoire, votre sens de l'observation, votre logique et votre vitesse de réaction. Le jeu peut se jouer en famille.

Pour installer *Brain Party*, saisissez les deux mots `brain party` dans la zone de recherche de l'outil d'ajout et de suppression de logiciels. Vous pouvez également demander l'installation depuis la ligne de commande, ce qui est plus rapide :

```
sudo apt-get install brainparty
```

L’icône du jeu sera disponible dans le menu des jeux, mais vous pouvez également le lancer directement sur la ligne de commande en saisissant ceci :

```
brainparty
```

Chapitre 20

Dix projets enthousiasmants

DANS CE CHAPITRE :

- **Trouver l'inspiration pour vous lancer**
- **Projets simples**
- **Projets plus ambitieux**

Si vous avez lu ce livre et réalisé les projets proposés, vous avez maintenant appris à programmer et à construire des montages électroniques avec le Raspberry Pi. Ce que vous allez découvrir et créer maintenant dépend de vous seul.

La variété de projets que produisent des gens de tous âges avec le Raspberry Pi est vraiment étonnante. Ce chapitre propose un bref tour d'horizon de quelques projets intéressants que nous avons découverts. Un lien est fourni pour chacun d'eux pour en savoir plus et construire le même projet ou s'en inspirer. Notez que les liens peuvent évoluer, auquel cas vous chercherez d'après le nom du créateur du projet associé à l'expression **Raspberry Pi**.

Un lecteur d'audiolivre à bouton unique

Michael Clemens a conçu à partir d'un Raspberry Pi un lecteur d'audiolivre pour sa grand-mère dont la vue basse empêche d'utiliser les minuscules boutons des baladeurs audio habituels.

Ce projet suppose un peu d'électronique : quelques transistors, une LED, une paire de haut-parleurs et un gros bouton dans un boîtier en plastique. Le bouton et la LED sont connectés à des broches GPIO du RasPi.

Un script Python sert à contrôler le logiciel de lecture avec le bouton. Un appui suspend et reprend la lecture du livre, un appui prolongé de quatre secondes fait revenir en arrière d'une piste.

Pour lire un autre audiolivre, il suffit d'insérer la clé qui le contient dans un port USB. Le contenu est d'office recopié sur le Raspberry Pi, en écrasant l'audiolivre précédent.

Vous trouverez des instructions, le code source Python et des photos sur le blog de Michael :

http://blogs.fsfe.org/clemens/2012/10/30/the-one-button-audiobook-player/

Une station météo

Quand Steve Wardell a entendu parler du Raspberry Pi, il a vite décidé que ce nano-ordinateur pourrait constituer la machine idéale pour se relier à sa station météo WS2350 en remplacement de son PC sous Windows. Un des défis à relever consistait à trouver comment se connecter à la station, qui offre un port série, au Raspberry Pi, qui offre un port USB. Après plusieurs essais, il a opté pour le circuit FT232RL.

Le RasPi collecte des données météorologiques de la station au moyen du paquetage d'outils Open2300, servant à lire des données. Steve a alors écrit un script Python pour convertir ces données afin de les poster sur Twitter. Le but ultime étant d'afficher des bulletins météo locaux sur son site Web.

Le blog de Steve explique en anglais comment il a réglé ses problèmes techniques pour intégrer les systèmes :

http://stevewardell.wordpress.com/tag/raspberrypi/

Un cardiogramme

www.raspberrypi.org/magpi/heartbeat-monitor

Daniel Fernandez a relié un appareil de suivi des battements du cœur à une carte Raspberry Pi, afin de pouvoir collecter ses données pendant ses séances de course sur tapis, en exportant les résultats dans un format standard pour analyse ultérieure. Le format proposé est le .csv, qui est récupérable directement par Excel et par LibreOffice.

Pour le capteur, il a choisi le modèle Polar H7, relié à une carte Raspberry Pi 3 équipée d'un écran de 8 cm, ce qui permet de bien voir les graphiques pendant l'effort. Il a réuni le tout dans un boîtier de protection.

Un skateboard électrique

www.youtube.com/watch?v=2WLEur3M8Yk

Vous pouvez vous servir d'une carte Raspberry Pi pour contrôler un skateboard motorisé, comme l'a prouvé un certain TheRaspberryPiGuy sur YouTube. Il a relié un moteur de type Alien Power Systems à son skateboard puis l'a fait contrôler par son Pi Zéro.

Pour contrôler la vitesse, il a utilisé une manette de console Nintendo Wii, en envoyant les commandes *via* Bluetooth. La vidéo le montre roulant à vive allure parmi les rues de Cambridge, qui est la ville de naissance de la fondation Raspberry Pi. Il arrive à atteindre les 30 km/h et estime l'autonomie à environ 10 km.

Le projet peut être enrichi d'un compteur de vitesse piloté par une carte RasPi, comme le montre le tutoriel suivant, qui indique également comment construire la carte :

www.instructables.com/id/The-Longboard-Speedometer

Un canon à T-shirts

http://drstrangelove.net/2014/01/raspberry-pi-powered-t-shirt-cannon

Les dénommés David Bryan et Lucas Saugen avaient été approchés par une équipe de patineuses à roulettes du Minnesota qui cherchaient comment réparer leur canon à T-shirts. Ce canon servait à envoyer des T-shirts dans le public pendant les pauses, qui durent environ 1 minute 30. Les patineuses leur ont demandé de trouver une solution pour pouvoir lancer plus d'un T-shirt pendant chaque pause, et prendre une photo du lancement pour la publier sur Twitter.

Le résultat combine quatre tubes transparents en PVC, avec une installation à air comprimé. Le canon possède trois boutons : un pour choisir un des quatre tubes et les deux autres à enfoncer ensemble pour lancer un T-shirt. Une clé est prévue comme mesure de sécurité

supplémentaire. L'appareil est contrôlé par une carte Raspberry Pi avec un logiciel écrit en Python exploitant la librairie GPIO.

Vous pouvez visiter le site de David pour en savoir plus sur la construction du canon, la récupération du code source et des vidéos.

Panflute hero

www.raspberrypi.org/archives/5924

Lors du Hackathon (une réunion de programmeurs) Way Out Ouest qui a eu lieu en 2013 à Göteborg en Suède, une équipe avait créé un jeu dans lequel il fallait souffler dans une flûte de pan en rythme avec la musique. Les flûtes étaient fabriquées avec quelques cannes de bambou dans lesquelles ils avaient ajouté des capteurs audio pour Arduino afin de détecter le souffle des joueurs. Les capteurs avaient été connectés à une carte Raspberry Pi, et exploités en Python avec la librairie RPi. GPIO. La carte Raspberry Pi était accrochée à l'arrière de la flûte de pan.

Le jeu était centralisé sur un PC de bureau sur lequel on faisait jouer des musiques pour flûte de pan récupérées sur le site Spotify. Les joueurs devaient souffler dans les tubes en rythme pour donner l'impression qu'ils jouaient les notes.

Magic Mirror

www.raspberrypi.org/archives/5275

À l'occasion de la semaine de la mode de Londres, la société Photobot.co avait créé un stand de photographie pour The Body Shop. Le stand était constitué de trois panneaux verticaux. Sur chaque panneau étaient installées cinq cartes Raspberry Pi avec autant de caméras. Le résultat constituait un miroir magique (Figure 20.1). Grâce aux multiples caméras, on obtenait un portrait en pied vu de gauche, de face et de droite.

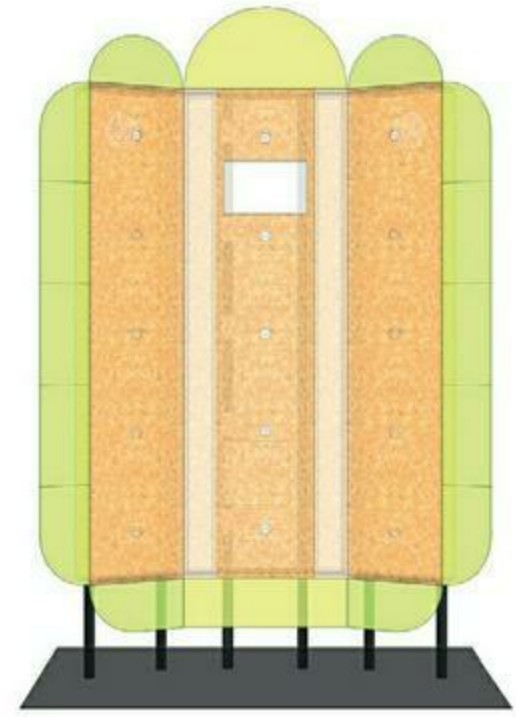


Figure 20.1 : Vue générale du stand Magic Mirror équipé de 15 cartes et de 15 caméras.

La prise de vue était déclenchée par l'envoi d'un tweet par le visiteur pour indiquer la couleur des vêtements qu'il portait. La photo après traitement montrait les trois angles de vue et était renvoyée par un tweet. La réponse comportait également une suggestion de couleur complémentaire, à condition que la couleur fournie par le visiteur ait été trouvée dans la centaine connue par le programme. Le système était basé sur un Mac mini. Les développeurs ont depuis fait remarquer qu'un Raspberry Pi était assez puissant pour faire le traitement lui-même. Un programme Python sur le contrôleur se chargeait de recueillir les tweets, de demander aux cartes Raspberry Pi de prendre les photos et de les transmettre, pour enfin créer l'image composite. Les connexions entre le contrôleur et les cartes RasPi étaient réalisées par liaison SSH, ce qui permettait un accès à distance aux images.

Pi in the sky

www.pi-in-the-sky.com

Dave Akerman a utilisé une carte RasPi pour contrôler des ballons stratosphériques gonflés à l'hydrogène. Ce genre de ballon monte suffisamment haut pour que les photos prises laissent voir la courbure du globe terrestre, avec le noir intersidéral au-dessus de l'atmosphère. Les

ballons montent pendant deux ou trois heures puis éclatent, ce qui permet à la charge utile de redescendre avec un parachute.

Une expérience marquante de Dave a été d'envoyer dans l'espace l'ours en peluche Babbage, qui est la mascotte de Raspberry Pi. Il est monté jusqu'à 39 km. Dave a avoué : « Le premier essai a échoué pour plusieurs raisons : trop de vent, ce qui a fait que la ligne de maintien de l'ours n'était pas au bon endroit pour que la résistance chauffante parvienne à la couper ; *secondo*, comme j'avais peur que l'ensemble soit trop lourd, j'avais choisi un jeu de piles trop petit ». « Mais la réussite est toujours plus agréable quand elle suit un échec, et j'étais très heureux quand j'ai su avec certitude que mon ours avait été relâché correctement lors du second essai ».

Parmi les projets précédents de Dave, nous pouvons citer la construction de systèmes de suivi GPS pour retransmettre des signaux de positionnement, des photos et des données vers le sol. Dorénavant, vous pouvez vous procurer le kit Pi in the sky, et profiter directement de toutes ces possibilités. Vous pouvez même combiner le kit avec la carte Sense HAT pour renvoyer les données par télétransmission. Dave s'est également servi du RasPi pour calculer le point d'atterrissage de la charge. C'est en effet un des grands défis des ballons stratosphériques que de récupérer la charge utile, car les calculs peuvent différer de la réalité d'environ 10 km.

Faire voler des ballons stratosphériques est une activité intéressante, mais elle réclame de la planification et de l'engagement. Vous devez obligatoirement demander une autorisation avant chaque vol pour ne pas perturber la circulation aérienne. Il faut également prévoir plusieurs centaines d'euros d'équipement, sans compter le ballon, l'hydrogène et le carburant pour aller récupérer la charge utile en voiture. Mais vous n'êtes pas seul : vous pouvez commencer par consulter le tutoriel de Dave à l'adresse suivante :

`www.daveakerman.com/?p=1732`

Voyez également les deux sites suivants :

`https://webchat.freenode.net` `www.ukhas.org.uk`

Le Turc mécanique

`www.raspberryturk.com`

Vous avez peut-être entendu parler du Turc mécanique, également appelé « joueur d'échecs automatique ». Il s'agissait d'un soi-disant automate

capable de jouer aux échecs. Il a fait illusion depuis son apparition en 1770 jusqu'à 1854. En réalité, il y avait un bon joueur d'échecs cachés sous la table, qui déplaçait les pièces et contrôlait la poupée habillée en turc. Ce turc mécanique a donné son nom à un service de travaux à la tâche d'Amazon qui consiste à rétribuer des gens pour travailler *via* Internet. Dorénavant, il existe aussi un Raspberry Turk, un vrai robot capable de jouer aux échecs.

Le projet Raspberry Turk repose sur le moteur de jeu d'échecs open source nommé Stockfish. La carte RasPi contrôle le bras qui déplace les pièces sur l'échiquier. Un module caméra Raspberry Pi est exploité avec un logiciel de vision artificielle pour savoir quand c'est à l'ordinateur de jouer ; le logiciel observe les mouvements de l'adversaire sur le plateau. Le tout est réuni dans une table avec l'échiquier peint sur le dessus.

Visitez le site Web du projet pour en savoir plus. Il combine de l'intelligence artificielle, de la robotique et de la vision synthétique.

Sound Fighter

www.foobarflies.io/pianette

Pour célébrer la réouverture de la Maison de la Radio à Paris en tant que centre culturel, Jean Weessa et Mélanie Pennec ont proposé de créer une installation basée sur deux pianos droits qui devaient servir de contrôleurs de jeu pour le jeu de combat *Street Fighter* sur PlayStation.

La réalisation avec un Raspberry Pi modèle B+ a été effectuée par Éric Redon et Cyril Chapellier. La carte recevait les frappes de touches grâce à des capteurs piézoélectriques installés sur les marteaux. Le projet a nécessité la création de circuits imprimés spécifiques reliés à une carte Arduino Uno pour envoyer les frappes de touches dans la console PlayStation. Le code source a été écrit en Python 3.

Le but était de faire en sorte que de bons pianistes deviennent de bons joueurs de ce jeu de baston, tout en donnant un résultat le plus musical possible. Les mouvements du personnage dans le jeu ont été affectés à la main gauche du pianiste et les actions de frappe à la main droite. Les deux pianos devaient jouer dans la même tonalité. Une modification du projet a permis d'utiliser les pédales pour simuler les appuis longs sur les boutons. Le projet a encore été enrichi en permettant de jouer également à deux autres jeux : *Tekken 5* et *Crash Nitro Kart*.

Visitez le site Web pour découvrir les nombreux défis qu'il a fallu relever. Vous y trouverez aussi le code source et les plans de construction.

Annexe A

Configurer et dépanner votre Raspberry Pi

DANS CE CHAPITRE :

- **Dépanner les problèmes les plus courants**
- **Retoucher la configuration du Raspberry Pi**
- **Monter manuellement un périphérique externe**
- **Corriger des soucis d'installation de logiciels**
- **Dépanner la connexion réseau**

La plupart des utilisateurs vont brancher leur Raspberry Pi, et tout va fonctionner comme sur des roulettes dès le premier jour. Si c'est votre cas, vous ne lirez peut-être jamais cette annexe !

Mais parfois, il y a un problème ou bien vous voudrez appliquer quelques retouches aux paramètres du nano-ordinateur, c'est-à-dire le reconfigurer.

Nous allons voir dans cette annexe comment régler les principaux problèmes recensés et comment intervenir sur certains paramètres. Vous n'aurez fort heureusement pas souvent besoin de venir lire cette annexe. Elle pourra s'avérer précieuse lorsque vous constaterez un comportement étrange lors du démarrage de la carte, ou si votre configuration est particulière.

Quoi que vous envisagiez de faire de particulier sur le Raspberry Pi (comme sur n'importe quel ordinateur d'ailleurs), il est conseillé de faire une sauvegarde d'abord (et périodiquement). En cas de blocage, vous pourrez tout réinstaller au lieu de pleurer parce que vous avez perdu des données précieuses en une fraction de seconde.

Dépanner le Raspberry Pi

Quand le coauteur Sean a démarré son Raspberry Pi la toute première fois, il n'a pas réussi à se connecter à Internet dans l'environnement graphique, même si tout se passait bien en mode texte sur la ligne de commande de l'interpréteur. En fait, le coupable s'est avéré être le clavier incompatible ! Jamais il n'aurait trouvé la solution en partant du symptôme. Voilà pourquoi nous vous conseillons de passer en revue dans l'ordre notre liste de contrôle en 12 étapes, quel que soit le problème apparent et même si nos contrôles semblent n'avoir aucun rapport avec le symptôme. Croyez-nous, vous serez agréablement surpris !

Ces étapes sont dans un ordre de priorités décroissantes, avec les tests et remèdes les plus évidents en premier. Vous pouvez faire chacun des contrôles à tout moment, mais en respectant à peu près notre ordre, vous gagnerez du temps et vous vous éviterez des interrogations.

N. d. T. : Nous nous permettons d'ajouter une première vérification qui corrige bon nombre de soucis : êtes-vous certain d'utiliser un **concentrateur/hub USB doté de sa propre alimentation** ? De nombreux problèmes aux conséquences étranges viennent soit d'un manque de puissance pour alimenter le RasPi par le hub, soit d'une trop grande consommation demandée sur les ports USB du RasPi.

1. **Soyez patient**.

   Quand votre Raspberry Pi est occupé, vous pourriez croire qu'il ne répond plus et qu'il est bloqué (planté). En général, il va refaire surface quand il en aura terminé avec son trop-plein de traitements. Si les actions en cours ne sont pas précieuses, vous pouvez toujours forcer le redémarrage ; vous perdez les données non sauvegardées. De plus, il est déconseillé de procéder avec ce genre de brutalité pendant une installation de logiciel (si vous pouvez l'éviter) parce que le processus se retrouve à moitié achevé. Rappelons que le RasPi possède un économiseur d'écran : si l'écran est noir, bougez la souris (dans l'environnement graphique). En mode

texte, utilisez une touche flèche du curseur ou la touche Maj pour le réveiller sans insérer de caractère.

2. **Redémarrez votre Raspberry Pi**.

   Il arrive rarement que la machine se bloque. Mais si cela arrive, mettez hors tension, attendez un peu et rallumez.

3. **Contrôlez toutes vos connexions**.

   Mettez le Raspberry Pi hors tension et vérifiez que tous vos câbles sont correctement insérés. Le Chapitre 3 présente les connexions entre le RasPi et tous les périphériques.

4. **Vérifiez que la carte mémoire SD est bien insérée**.

   Regardez le circuit du RasPi de près. Si la LED rouge PWR est allumée, mais pas la LED verte OK, c'est que le RasPi a des soucis pour exploiter la carte SD. Vérifiez d'abord qu'elle est bien insérée après avoir éteint (voir le Chapitre 3).

5. **Essayez avec une autre carte SD**.

   Si la diode rouge s'allume mais pas la verte, essayez avec une autre carte mémoire. Il nous est arrivé de rencontrer des soucis avec certaines cartes SD, parfois à cause de l'adaptateur en plastique qui permet d'utiliser une carte microSD dans un lecteur SD. Vous pouvez consulter la liste des modèles de cartes qui ont été considérés comme compatibles avec les Raspberry Pi à l'adresse suivante : `http://elinux.org/RPi_SD_cards`.

6. **Déconnectez les périphériques**.

   Essayez de déconnecter le concentrateur/hub USB, le clavier et la souris puis redémarrez. Bien sûr, cela ne sert à rien si le problème concerne l'accès à un périphérique ou requiert sa présence pour être reproduit, mais cela peut vous aider à détecter une incompatibilité de matériel qui empêche le RasPi de démarrer normalement. S'il vous faut un clavier, branchez-le directement sur un port USB du RasPi. Vous le débrancherez après avoir saisi le mot de passe ou lancé le programme qui pose problème. Si le RasPi fonctionne normalement sans le ou les périphériques, avancez par élimination progressive (branchez les périphériques un par un en redémarrant) pour trouver le coupable.

7. **Essayez avec d'autres périphériques**.

   Si possible, essayez un autre clavier, souris et concentrateur/hub, en privilégiant ceux indiqués dans la page http://elinux.org/RPi_VerifiedPeripherals La majorité des soucis rencontrés ont trait à un périphérique incompatible avec le Raspberry Pi. Remplacer le clavier, puis la souris, puis le concentrateur USB peut résoudre un problème (y compris le problème d'accès à Internet mentionné plus haut). L'étape précédente vous aidera à trouver quel est le périphérique coupable.

8. **Changez les câbles**.

Les câbles défectueux ou de mauvaise qualité concernent surtout la liaison Ethernet, l'écran et la sortie audio.

9. **Essayez un autre écran**.

   Si rien ne s'affiche, alors que le RasPi semble avoir démarré (la LED rouge brille et la verte scintille), branchez un autre écran ou un téléviseur. Le Chapitre 3 explique comment.

10. **Mettez vos logiciels à jour**.

    Il faut bien sûr que votre connexion Internet fonctionne. Si c'est le cas, demandez une mise à jour du système et de tous les logiciels installés actuellement (sans crainte d'effacer vos données) au moyen de cet excellent enchaînement de commandes Linux :

    ```
    sudo apt-get update & & sudo apt-get upgrade
    ```

11. **Essayez une autre alimentation secteur**.

    Nous indiquons cette étape vers la fin parce qu'elle suppose de trouver une autre alimentation ou d'en acquérir une. Pourtant, les faiblesses des alimentations représentent une des causes majeures de problèmes aussi variés qu'étonnants. Si vous avez un collègue dont le Raspberry Pi fonctionne bien, empruntez-lui son alimentation. Rappelons que le Raspberry Pi 3 consomme plus que les modèles précédents. Si vous avez acheté ce nouveau modèle, mais conservé votre alimentation antérieure, vous risquez d'avoir des soucis. Dans le bureau graphique PIXEL, il y a une

icône avec un éclair dans le coin supérieur droit lorsque la carte est sous-alimentée et une icône de thermomètre lorsqu'elle chauffe trop. Nous avons indiqué dans le Chapitre 1 quels critères satisfaire pour acheter une alimentation.

12. **Cherchez une solution sur le Web**.

    Nous ne pouvons pas décrire toutes les éventualités ici. Si le problème n'est pas résolu, lisez la suite de cette annexe puis cherchez sur les sites de plus en plus nombreux qui parlent du Raspberry Pi. Les anglophones iront lire la page `http://elinux.org/R-Pi_Troubleshooting` et les forums sur `www.raspberrypi.org`. Vous n'êtes sans doute pas le seul à subir ce problème.

Ajuster les paramètres système

Les paramètres de fonctionnement de votre Raspberry Pi sont stockés sur la carte SD, la plupart sont rassemblés dans le fichier nommé `config.txt` qui se trouve dans le répertoire `/boot`. Vous êtes autorisé à modifier ce fichier, et vous pouvez le faire avec un outil éditeur de texte minimal portant le nom Nano et préinstallé dans votre distribution du système Linux.

Avant d'intervenir manuellement dans les paramètres, rappelons que vous pouvez épuiser d'abord toutes les possibilités qu'offre l'outil de configuration de PIXEL ou l'outil **raspi-config** qui permet *via* un menu d'accéder aux principaux paramètres (voir le Chapitre 3). Vous pouvez lancer cet outil à tout moment en écrivant dans l'interpréteur :

```
sudo raspi-config
```

L'outil **raspi-config** permet les opérations suivantes :

- **Configuration du clavier.** Dans les options d'internationalisation, vous pouvez changer le type de clavier. Voyez également les options de

localisation de l'outil de configuration dans l'interface graphique. Nous en avons parlé dans le Chapitre 3.

» **Problèmes de caméra.** Vous devez vérifier que la caméra est activée dans l'outil de configuration. Vous pouvez l'activer par la page **Interfaces** de l'outil de configuration graphique.

» **Problèmes de son.** Dans les options avancées, vous pouvez forcer l'envoi des signaux audio soit vers la prise casque, soit vers la sortie HDMI.

» **Problème d'espace libre sur la carte SD.** Si vous avez implanté le système sur la carte en partant d'une image, voyez si vous pouvez augmenter l'espace disponible pour utiliser tout l'espace libre. Voyez l'option concernant le système de fichiers dans les options avancées.

L'interpréteur a été présenté en détail dans le Chapitre 5. Pour l'essentiel, rappelons que c'est face à lui que vous vous trouvez au démarrage juste après avoir réussi à fournir le mot de passe de votre compte utilisateur sur le Raspberry Pi. Vous y accédez aussi en ouvrant une fenêtre LX Terminal *via* son icône sur le bureau (Chapitre 5).

L'outil **raspi-config** et son équivalent dans PIXEL savent apporter les retouches aux fichiers de configuration sans vous demander d'intervenir dans les fichiers tels que *config.txt*, ce qui réduit les risques d'erreur. Mais si l'option qui vous intéresse n'est pas gérée par les menus, il va vous falloir plonger dans le fichier de configuration.

Avant de toucher au fichier *config.txt*, vérifiez que vous avez sauvegardé vos données personnelles (voyez si nécessaire plus loin comment monter manuellement un périphérique de stockage externe). Une erreur de paramétrage peut par exemple désactiver l'affichage ; vous auriez ensuite du mal à récupérer vos fichiers en aveugle. La seule solution serait alors de réparer le fichier *config.txt* sur un autre ordinateur ou plus simplement de l'effacer de la carte SD. Dans ce cas, au prochain redémarrage sur le Raspberry Pi, le circuit va utiliser les paramètres par défaut (paramètres usine). C'est une bonne solution à condition que le Raspberry Pi

fonctionne avec ces paramètres au vu des périphériques connectés.

L’éditeur Nano pour modifier config.txt

Pour charger le fichier *config.txt* dans l’éditeur Nano, saisissez la commande suivante dans l’interpréteur, tout en minuscules :

```
sudo nano /boot/config.txt
```

La figure suivante montre l’éditeur Nano avec le fichier *config.txt* chargé (Figure A.1).

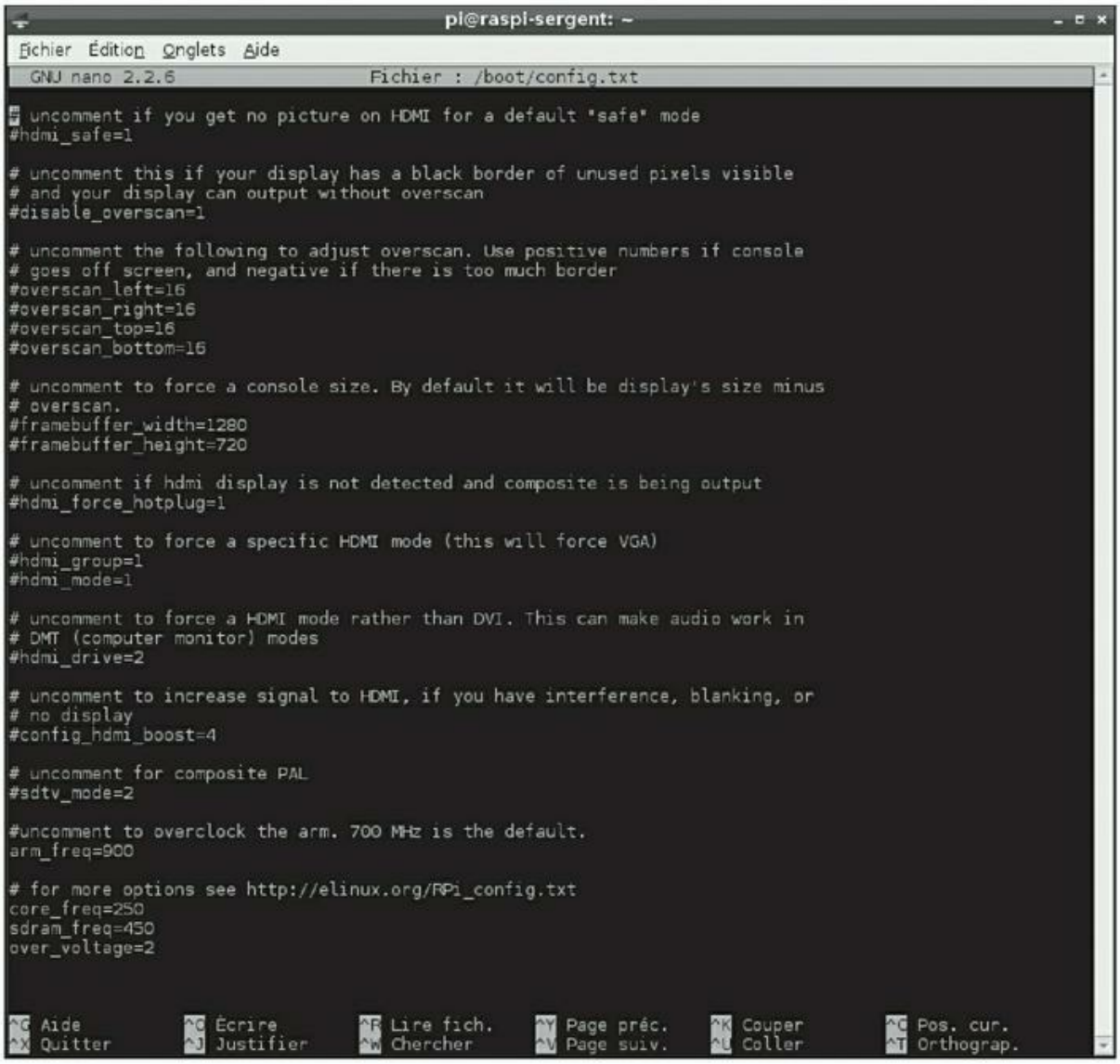

pi@raspi-sergent: ~
Fichier Édition Onglets Aide
GNU nano 2.2.6 Fichier : /boot/config.txt
uncomment if you get no picture on HDMI for a default "safe" mode
#hdmi_safe=1
uncomment this if your display has a black border of unused pixels visible
and your display can output without overscan
#disable_overscan=1
uncomment the following to adjust overscan. Use positive numbers if console
goes off screen, and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16
uncomment to force a console size. By default it will be display's size minus
overscan.
#framebuffer_width=1280
#framebuffer_height=720
uncomment if hdmi display is not detected and composite is being output
#hdmi_force_hotplug=1
uncomment to force a specific HDMI mode (this will force VGA)
#hdmi_group=1
#hdmi_mode=1
uncomment to force a HDMI mode rather than DVI. This can make audio work in
DMT (computer monitor) modes
#hdmi_drive=2
uncomment to increase signal to HDMI, if you have interference, blanking, or
no display
#config_hdmi_boost=4
uncomment for composite PAL
#sdtv_mode=2
#uncomment to overclock the arm. 700 MHz is the default.
arm_freq=900
# for more options see http://elinux.org/RPi_config.txt
core_freq=250
sdram_freq=450
over_voltage=2
^G Aide ^O Écrire ^R Lire fich. ^Y Page préc. ^K Couper ^C Pos. cur.
^X Quitter ^J Justifier ^W Chercher ^V Page suiv. ^U Coller ^T Orthograp.

Figure A.1 : Vue générale de l’éditeur Nano avec config.txt ouvert.

Vous vous déplacez dans le document avec les touches fléchées du curseur. Le bas de fenêtre rappelle les principaux raccourcis clavier pour

contrôler Nano. L'accent circonflexe représente la touche **Ctrl**. Les raccourcis peuvent vous sembler inhabituels. Voici les principaux :

- **Ctrl + W** : Recherche de mot ou de phrase. La lettre d'option se mémorise avec *Where* (Où ?). Permet de trouver l'option de configuration à modifier.
- **Ctrl + V** : Page-écran suivante. Au départ, le fichier *config.txt* n'occupe pas plus d'un écran, mais cette commande vous sera utile pour éditer d'autres fichiers avec Nano.
- **Ctrl + Y** : Page-écran précédente.
- **Ctrl + K** : Coupe la ligne de texte courante.
- **Ctrl + U** : Colle la ligne coupée dans la ligne courante du curseur.
- **Ctrl + G** : Affiche l'aide sur les commandes de Nano.
- **Ctrl + O** : Enregistre le fichier modifié.
- **Ctrl + X** : Quitte Nano et ramène au niveau de l'interpréteur.

Ce que vous remarquez d'abord est la présence du signe de neutralisation # au début de toutes les lignes dans *config.txt*. Ce symbole demande au système d'ignorer tout ce qui le suit jusqu'à la fin de la ligne. Si vous vous demandez à quoi sert d'écrire des lignes inutiles, sachez que cette pratique est très répandue (pas assez selon certains) pour aider les humains à comprendre le contenu d'un fichier technique (code source ou fichier de configuration). Les deux utilisations de ce signe sont visibles ici : pour indiquer des commentaires destinés aux humains et pour maintenir inactive une instruction qu'il devient très simple d'activer simplement en ôtant le signe #.

Voici les deux premières lignes du fichier *config.txt* (N.d.T. : nous en profitons pour traduire) :

```
# Décommentez si pas d'image HDMI pour activer le
mode "safe"
#hdmi_safe=1
```

La première ligne est un commentaire pour vous, mais la deuxième est un paramètre désactivé pour l'instant. Il permet d'activer le mode sécurisé du HDMI (*safe mode*). Tous les autres paramètres de *config.txt* se présentent ainsi :

```
nom_param=valeur
```

Chaque paramètre est sur une ligne distincte. Pour activer le mode sécurisé du HDMI, il suffit de supprimer le préfixe de commentaire de la deuxième ligne (de la « décommenter »). Les deux premières lignes doivent ensuite se lire ainsi :

```
# uncomment if you get no picture on HDMI for a
default "safe" mode
hdmi_safe=1
```

N'enlevez pas le signe # de la ligne de commentaires qui est absolument incompréhensible pour l'ordinateur. Ne supprimez que celui de la deuxième ligne.

La simple suppression d'un caractère fait toute la différence ! Enregistrez le fichier par Ctrl + O puis quittez Nano par Ctrl + X et redémarrez le RasPi pour activer le mode. Voici comment redémarrer à chaud (sans mettre hors tension) :

```
sudo reboot
```

Pour désactiver un paramètre actuellement en vigueur, il suffit de rajouter un symbole # en début de ligne et de redémarrer.

Vous pouvez ajouter des lignes pour vos propres commentaires. Nous conseillons par exemple d'ajouter au début une ou plusieurs lignes (un dièse au début de chacune !) pour expliquer quelles modifications vous avez apportées avec la date, ce qui vous aidera au dépannage ou pour restaurer les options plus tard.

Dépannage de l'affichage

Le Raspberry Pi s'adapte à une grande variété d'écrans et de téléviseurs, mais cela implique qu'il faille dans certains cas retoucher les réglages du RasPi pour les préciser, aller à l'encontre d'un réglage actuel ou changer une préférence.

Dès que vous avez un souci de stabilité ou de lisibilité de l'image, parcourez le Tableau A.1.

Vous pouvez modifier plusieurs paramètres à la fois, mais chacun doit être spécifié sur une ligne indépendante. La plupart sont déjà présents dans le fichier *config.txt*, mais n'oubliez pas d'enlever le signe # qui sert à neutraliser ce qui le suit.

Nous supposons ici que l'écran est allumé, connecté et correctement réglé au niveau de ses propres paramètres. Vérifiez tout cela avant d'intervenir sur la configuration d'affichage du RasPi.

Tableau A.1 : Résolution des problèmes d'affichage.

Symptôme	Paramètre concerné	Valeur
Portion d'image perdue par la gauche	`overscan_left`	Les paramètres `overscan` sont exprimés en pixels d'écran. Une valeur négative réduit le débord, une positive l'augmente. Exemple : `overscan_left=50`.
Portion d'image perdue par la droite	`overscan_right`	Idem du côté droit : `overscan_right=50`.
Portion d'image perdue par le haut	`overscan_top`	Idem en haut : `overscan_top=50`.
Portion d'image perdue par le bas	`overscan_bottom`	Idem en bas : `overscan_bottom=50`.

Cadre noir tout autour	`disable_overscan`	La valeur 1 inhibe le surbalayage : `disable_overscan=1`.
Texte et contenus trop petits	`framebuffer_width`, `framebuffer_height`	Voir la section qui suit ce tableau.
Texte et contenus trop gros	`framebuffer_width`, `framebuffer_height`	Voir la section qui suit ce tableau.
Erreur d'affichage sur écran téléviseur à écran cathodique	`sdtv_mode`	La sortie vidéo composite est en NTSC par défaut (mode Amérique du Nord). Choisir 0 pour NTSC, 1 pour NTSC-J (Japon), 2 pour le PAL (Europe) ou 3 pour le PAL-M (Brésil). Pour la France, décommentez `sdtv_mode=2`.
Image déformée sur téléviseur Mauvaises proportions	`sdtv_aspect`	Le rapport d'aspect relie la largeur et la hauteur de l'image. Trois choix sont possibles : 1 (rapport 4 : 3), 2 (rapport 14 : 9) et 3 (rapport 16 : 9). Exemple : `sdtv_aspect=2`.
Écran HDMI vide	`hdmi_force_hotplug`	Si le RasPi ne détecte pas l'écran HDMI, vous le forcez à afficher sur la sortie HDMI en donnant la valeur 1 à ce paramètre, comme ceci : `hdmi_force_hotplug=1`.
	`hdmi_drive`	

Écran DVI neigeux ou distordu		Permet de jouer sur la tension fournie sur le port HDMI. Avec un écran DVI, essayez la valeur 1. Avec un écran HDMI, essayez 2. Exemple : `hdmi_drive=1`.
Pas de son sur l'écran	`hdmi_drive`	Forcez le mode HDMI avec la valeur 2, ce qui envoie le signal audio sur le HDMI. Exemple : `hdmi_drive=2`.
Image vide ou parasitée	`config_hdmi_boost`	Les valeurs autorisées vont de 1 à 7 et décident de la puissance du signal sur le port HDMI. Augmentez la valeur étape par étape. Le mode sécurisé HDMI utilise par exemple la valeur 4. Exemple : `config_hdmi_boost=4`.
Autre problème avec un écran HDMI	`hdmi_safe`	En cas de souci sur écran HDMI, testez le mode sécurisé (*safe*) qui regroupe plusieurs des choix précédents : `hdmi_force_hotplug=1`, `config_hdmi_boost=4`, `disable_overscan=1` et valeurs sûres pour `hdmi_mode` et `hdmi_group`. Redéfinit les réglages de résolution HDMI. Pour tous détails, http://elinux.org/RPi_config.txt. Exemple : `hdmi_safe=1`.

Ajustement des dimensions de l'écran

Vous pouvez intervenir sur le nombre de lignes et de colonnes de pixels de l'affichage. Si vous réduisez les valeurs, le contenu de l'image sera grossi. Les deux paramètres appropriés se nomment `frame buffer_width` et `framebuffer_ height`. Voici par exemple comment réduire l'affichage à 1 024 lignes sur 768 colonnes :

```
framebuffer_width=1024
framebuffer_height=768
```

Le fichier *config.txt* contient déjà ces paramètres, mais neutralisés par le symbole de commentaires. Supprimez ce signe # puis modifiez les valeurs selon vos besoins.

Vous pouvez également modifier ces paramètres depuis le bureau graphique grâce à l'outil de configuration décrit dans le Chapitre 3. Choisissez l'option pour définir la résolution.

Autres paramètres

Bien d'autres paramètres sont disponibles pour le Raspberry Pi dans son fichier *config.txt*, mais la place nous manque pour les décrire. Voyez la page suivante : `http://elinux.org/RPi_config.txt`.

Correction des problèmes de son

Dans l'environnement graphique, vous disposez en haut à droite d'une icône Haut-parleur pour régler le volume sonore ou couper le son. Vous pouvez également travailler sur ligne de commande en démarrant l'outil suivant :

```
sudo alsamixer
```

Vous ajustez le volume avec les flèches du clavier ou la molette de la souris.

Le périphérique de sortie audio est détecté automatiquement. Si vous envoyez l'affichage vers un écran HDMI, le signal audio est envoyé sur la même sortie, même si l'écran n'a pas de haut-parleur. Il faut dans ce cas

utiliser les options de configuration pour rediriger le signal audio vers la prise casque. Dans le système multimédia LibreELEC (décrit dans le Chapitre 8), vous disposez de l'icône des paramètres en haut à gauche. Entrez dans les paramètres système pour accéder à la section audio. C'est ici que vous pourrez changer le périphérique de sortie.

Montage manuel de périphériques amovibles

Quand vous branchez un périphérique de stockage externe (clé ou disque dur USB), il est reconnu automatiquement quand vous êtes dans l'environnement graphique et une fenêtre s'ouvre pour vous demander confirmation. Mais cela ne se produit pas ainsi dans l'interpréteur en mode texte. C'est à vous d'émettre une commande pour raccorder logiquement le périphérique au système de fichiers, c'est-à-dire le *monter* dans un répertoire depuis lequel le contenu du support externe deviendra accessible.

Si vous n'avez besoin que de copier des fichiers sur ce support externe en guise de sauvegarde, il sera plus rapide de revenir dans l'interface graphique pour profiter du montage automatique et du confort du Gestionnaire de fichiers (Chapitre 4).

Pour monter un périphérique, il faut d'abord créer un répertoire d'accueil qui deviendra son *point de montage*. Pour voir le contenu du périphérique, il suffira d'entrer dans ce répertoire. Ce dernier est réutilisable : vous pouvez démonter un disque externe, puis monter un autre disque au même endroit. Le sous-répertoire peut être créé où bon vous semble (même dans votre répertoire personnel), mais une pratique très répandue consiste à monter les supports temporaires dans le répertoire prédéfini à cet effet puisqu'il a été nommé `/mnt` :

```
sudo mkdir /mnt/diskusb
```

Nous devons maintenant nous renseigner sur le périphérique. Insérez-le pour que le système le détecte à bas niveau puis saisissez cette commande :

```
sudo fdisk -l
```

Le caractère de l'option est bien la lettre **l** minuscule, et non le chiffre **1**. Voici un exemple d'affichage résultant :

```
Disk /dev/mmcblk0: 4025 MB, 4025483264 bytes
4 heads, 16 sectors/track, 122848 cylinders,
total 7862272 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512
bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000714e9

  Device Boot       Start     End   Blocks  Id
Système
/dev/mmcblk0p1        8192  122879    57344   c
W95 FAT32 (LBA)
/dev/mmcblk0p2     122880 7862271  3869696  83
Linux

Disk /dev/sda: 16.0 GB, 16037969920 bytes
32 heads, 63 sectors/track, 15537 cylinders,
total 31324160 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512
bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x1707001e

  Device Boot  Start        End    Blocks   Id
Système
/dev/sda1 *       63   31322591  15661264 +    c
W95 FAT32 (LBA)
```

Vous obtenez la liste des périphériques de stockage actuellement détectés par le RasPi. Dans l'exemple, le premier disque (*Disk /dev/mmcblk0*) offre 4025 Mo, c'est donc une carte mémoire SD de 4 Go. L'autre (*Disk /dev/sda*) de 16 Go est une clé USB venant d'être insérée. Ce qui nous intéresse, c'est le nom de périphérique/numéro de partition, que nous pouvons lire en bas : `sda1`.

Pour monter cette partition sda1 pour l'utilisateur *pi* (`uid=pi`) et le groupe *pi* (`gid=pi`), nous écrivons ceci :

```
sudo mount -o uid=pi,gid=pi /dev/sda1
/mnt/diskusb
```

Vous pouvez maintenant accéder à la clé USB, par exemple lister son contenu. Le répertoire racine de la clé correspond dorénavant au sous-répertoire `diskusb` :

```
ls /mnt/diskusb
```

Pour créer une copie de sauvegarde de tout ce que contient votre répertoire personnel vers la clé USB, vous écrivez :

```
cp –R ~/* /mnt/diskusb
```

Correction d'erreurs d'installation de logiciels

L'outil **apt** réussit normalement à installer et supprimer les logiciels correctement. Si un logiciel refuse de fonctionner, essayez d'abord de le désinstaller puis de le réinstaller comme montré ans le Chapitre 5.

La plupart des paquetages d'applications dépendent de la présence d'autres paquetages (ce sont leurs dépendances). Le gestionnaire de paquetages les gère, mais en cas d'altération dans les dépendances, vous pouvez tenter de réparer l'ensemble ainsi :

```
sudo apt-get -f install
```

Dépannage de la connexion réseau

Dans l'environnement graphique (Chapitre 4), vous pouvez aisément vérifier si votre connexion à Internet fonctionne en lançant le navigateur Web. Dans l'interpréteur shell, servez-vous de la commande d'interrogation `ping` :

```
ping –c 5 www.google.com
```

Cet exemple demande à cinq reprises de se connecter au site de Google puis affiche les résultats. Vous devez voir cinq paquets de données transmis et reçus avec un temps total. Si cela échoue, essayez un autre site, au cas peu probable où le site de Google soit surchargé. Il arrive qu'un logiciel pare-feu gêne la commande `ping`, mais c'est rare. Si les résultats sont corrects, vous êtes assuré que le RasPi peut se connecter à Internet.

Voici comment obtenir la liste des périphériques réseau détectés par le RasPi :

```
ifconfig
```

Vous devez voir deux sections. La première concerne la carte réseau *eth0* (votre connexion Ethernet) et/ou la carte WiFi *wlan0* et l'autre la boucle locale *lo* (*loopback*), qui permet à l'ordinateur de faire communiquer des processus internes au Raspberry Pi et que vous pouvez ignorer ici. S'il y a une adresse `inet addr` (pas `inet6 addr`) pour *eth0* ou *wlan0*, c'est la preuve que votre RasPi a pu se connecter à un routeur et qu'il a obtenu une adresse IP.

La connexion Ethernet devrait s'activer automatiquement, mais si cela n'est pas le cas, essayez de l'activer manuellement ainsi :

```
sudo ifup eth0
```

De même, pour la désactiver :

```
sudo ifdown eth0
```

Le RasPi se connecte de lui-même à un routeur grâce au protocole DHCP (*Dynamic Host Configuration Protocol*). Vérifiez le cas échéant dans les paramètres de votre routeur (ou de votre boîtier Internet) que les fonctions routeur et DHCP sont bien activées.

En cas de soucis réseau, essayez aussi de changer le câble Ethernet et voyez si l'alimentation de votre RasPi fournit assez de courant au circuit (revoyez le Chapitre 1).

Revoyez le Chapitre 3 si nécessaire pour configurer votre connexion Wi-Fi.

Connexion sécurisée SSH

Si votre carte RasPi est dotée d'une connexion réseau, vous devez pouvoir y accéder à distance depuis une autre machine du réseau local grâce à l'outil SSH (*Secure Shell*). Cet outil d'accès à distance permet d'utiliser une carte Raspberry Pi sans écran (*headless*). Voici comment configurer cette connexion :

1. **Commencez par changer votre mot de passe**.

   Mieux vaut changer le mot de passe avant de rendre votre Raspberry Pi accessible à distance. Servez-vous de l'outil de configuration ou reportez-vous au Chapitre 5 qui explique comment modifier un mot de passe depuis la ligne de commande.

2. **Activez la fonction SSH dans le système Raspbian**.

   Toujours avec l'outil de configuration, vous activez SSH dans la page **Interfaces**. Vous pouvez également avec la commande suivante créer un fichier vide portant le nom *ssh* dans le répertoire nommé *boot* puis redémarrer votre carte :

```
sudo touch /boot/ssh
```

3. **Récupérez l'adresse IP de votre carte**.

   Sur la ligne de commande, saisissez la commande **ifconfig** pour afficher l'adresse IP (voyez la valeur de **inet addr**).

4. **Connectez-vous à distance**.

   Sur un PC sous Windows, vous devez télécharger et installer un outil client SSH, comme par exemple **Putty**. Dans cet outil, vous saisissez l'adresse IP de la carte et vous cliquez le bouton **Open**. Vous avez alors accès à la ligne de commande de la carte Raspberry Pi. Vous pouvez gérer les fichiers, les

afficher sur l’écran de votre PC, le tout en utilisant le clavier de votre PC. Il existe même des logiciels clients pour se connecter depuis un iPhone ou une machine sous Android. Le Mac n’a pas besoin d’outil complémentaire, car il accède directement par l’application **Terminal**. Pour d’autres instructions au sujet de SSH, voyez la page suivante :

www.raspberrypi.org/documentation/remote-access/ssh

Sachez que vous n’avez accès qu’à la ligne de commande avec SSH, pas au bureau graphique.

Sommaire

Couverture

Raspberry Pi pour les Nuls grand format, 2e édition

Copyright

Introduction

- À propos de ce livre
- En quoi ce livre vous sera utile ?
- Quelques conditions préalables
- Icônes utilisées
- Le fichier archive des exemples
- Par où commencer ?

PARTIE 1. Premiers pas avec le Raspberry Pi

- Chapitre 1. Présentation du Raspberry Pi
  - Premier contact avec le Raspberry Pi
  - Possibilités du Raspberry Pi
  - Se procurer un Raspberry Pi
  - Accessoires indispensables
- Chapitre 2. Télécharger le système d’exploitation
  - Découverte de Linux
  - Préparation d’une carte NOOBS
  - Utiliser sa carte NOOBS
  - Créer une carte SD système
- Chapitre 3. Connecter votre Raspberry Pi
  - Insertion de la carte SD

Branchement du module caméra Raspberry Pi

Adaptation des Pi Zéro et Pi Zéro W

Branchement d’un écran ou d’un téléviseur

Branchement d’une multiprise USB (hub)

Branchement d’un clavier et d’une souris

Branchements audio

Branchement au routeur

Branchement de l’alimentation

Démarrage

Ouverture de session (login)

Configuration du système Raspbian

Configuration Wi-Fi

Configuration Bluetooth

Test du module Caméra

Configurer la partition data

Prêt pour l’aventure ?

PARTIE 2. Découvrir Linux

Chapitre 4. L’environnement graphique

Naviguer sur le bureau

Le Gestionnaire de tâches

Le Gestionnaire de fichiers

Naviguer sur le Web

Protection de la vie privée

La messagerie Claws Mail

Le Visionneur d’images

L’éditeur de texte

Personnaliser le bureau graphique

Ajouter et enlever des applications

Sauvegarde des données

Déconnexion et arrêt système

Chapitre 5. L’interpréteur Linux shell

L’invite de commande

Exploration du système Linux

Optimiser la saisie de commandes

Supprimer un répertoire (rmdir, rm)

Installer et gérer les logiciels du Raspberry Pi

Se documenter sur les commandes

Définir une nouvelle commande Linux

Arrêter et redémarrer le système

PARTIE 3. Travailler et se cultiver avec le Raspberry Pi

Chapitre 6. La bureautique avec le Raspberry Pi

Installation de LibreOffice sur le Raspberry Pi

Démarrage de LibreOffice

Sauvegarder le travail

Rédiger un courrier avec Writer

Gérer un budget avec Calc

Créer un diaporama avec Impress

Créer une invitation avec Draw

Chapitre 7. Retouches photos avec GIMP

Installer et démarrer GIMP

La fenêtre de travail de GIMP

Redimensionner une image

Rogner une photo

Rotation et miroir

Corriger les couleurs

Corriger les imperfections

Exporter une image dans différents formats

Pour en savoir plus au sujet de GIMP

Chapitre 8. Le Raspberry Pi comme centre multimédia

Préparation du centre multimédia

Découverte du centre multimédia

Le multimédia dans Raspbian

PARTIE 4. Apprendre à programmer avec le Raspberry Pi

Chapitre 9. Introduction à la programmation visuelle avec Scratch

Programmer, c’est quoi ?

Scratch 1 ou Scratch 2 ?

La fenêtre de l’atelier Scratch

Positionner et redimensionner un objet

Déplacer l’objet par programme

Modifier l’aspect d’un objet

Effets sonores et musique

Créer un script

Ralentir un mouvement avec le bloc Attendre

Sauvegarder son travail

Nouveautés de Scratch 2

Chapitre 10. Programmer un jeu d’arcade en Scratch

Lancement d’un nouveau projet Scratch

Modifier l’arrière-plan

Ajouter des objets visuels

Dessiner un objet visuel dans Scratch

Nommer les objets

Contrôler l’exécution des scripts

Exploiter des valeurs aléatoires

Détecter une collision entre deux objets

Utilisation de variables

Déplacer automatiquement un objet

Détecter et corriger un bogue

Ajouter un script pour la scène

Dupliquer un objet

Tester le jeu

Régler la vitesse de jeu

Pour aller plus loin

Chapitre 11. Programmez en Python

Démarrer Python

Saisie de commandes Python

Quelques calculs en Python

Un programme de tables de multiplication

Création du projet Chatbot

Chapitre 12. Un jeu avec Python et Pygame Zero

Récupérer des images et des sons

Écrire et exécuter un premier programme

Listing complet du code source

Pour aller plus loin dans Pygame Zero

Chapitre 13. Programmer Minecraft en Python

Découvrir le monde Minecraft

Préparer Python pour Minecraft

Définir les paramètres du labyrinthe

Chapitre 14. De la musique avec Sonic Pi

L’interface visuelle de Sonic Pi

Vos premières notes

Noms des notes et accords

Pour alléger l’écriture

Une mélodie aléatoire avec Shuffle

Au gré du hasard

Nommer vos listes

Faire jouer des hauteurs au hasard

Le temps réel avec live_loop

Reproduire des échantillons

Ajouter des effets spéciaux (Fx)

Synchronisation avec une piste rythmique

Récapitulons

Plus loin avec Sonic Pi

PARTIE 5. Découvrir l’électronique avec le Raspberry Pi

Chapitre 15. Principes des circuits électroniques et soudure

Qu’est-ce qu’un circuit électrique ?

Le connecteur GPIO

Découverte du fer à souder

Cartes d’extensions du commerce

La carte Gertboard

Autres cartes d’extension

Quelques cartes d’extension

Chapitre 16. Contrôler les circuits du RasPi

Accès aux broches GPIO

Votre premier circuit

Clignotement de LED avec Python

Construire un dé électronique

Simulateur de passage piéton

Chapitre 17. Utiliser des diodes LEDRGB

Des LED de toutes les couleurs

Mélangeons les couleurs

Des LED intelligentes

Les sept envahisseurs de l’arc-en-ciel

Le pied jongleur *Keepy Uppy*

Des galaxies de LED

Chapitre 18. L’identification par radiofréquences

Principe du RFID

Un premier juke-box RFID

Un juke-box RFID plus versatile

Pour aller plus loin

PARTIE 6. Les Dix Commandements

Chapitre 19. Dix logiciels sélectionnés pour le Raspberry Pi

Penguins Puzzle

FocusWriter

Mathematica

Invaders 3D

Fraqtive

Tux Paint

Grisbi

Beneath a Steel Sky

Sense HAT Emulator

Brain Party

Chapitre 20. Dix projets enthousiasmants

Un lecteur d’audiolivre à bouton unique

Une station météo

Un cardiogramme

Un skateboard électrique

Un canon à T-shirts

Panflute hero

Magic Mirror

Pi in the sky

Le Turc mécanique

Sound Fighter

Annexe A. Configurer et dépanner votre Raspberry Pi

Dépanner le Raspberry Pi

Ajuster les paramètres système

Correction des problèmes de son

Montage manuel de périphériques amovibles

Correction d’erreurs d’installation de logiciels

Dépannage de la connexion réseau

Connexion sécurisée SSH

Table of Contents

Raspberry Pi pour les Nuls grand format, 2e édition 2
Copyright 3
Introduction 4
- À propos de ce livre 4
- En quoi ce livre vous sera utile ? 5
- Quelques conditions préalables 6
- Icônes utilisées 7
- Le fichier archive des exemples 8
- Par où commencer ? 9

PARTIE 1. Premiers pas avec le Raspberry Pi 10
- Chapitre 1. Présentation du Raspberry Pi 11
  - Premier contact avec le Raspberry Pi 13
  - Possibilités du Raspberry Pi 18
  - Se procurer un Raspberry Pi 19
  - Accessoires indispensables 19
- Chapitre 2. Télécharger le système d’exploitation 30
  - Découverte de Linux 31
  - Préparation d’une carte NOOBS 32
    - Télécharger NOOBS 33
    - Formater la carte mémoire SD 33
      - Formatage sous Windows 34
      - Formatage sous macOS 35
      - Formatage sous Linux 36
    - Copie de NOOBS vers la carte SD 39
  - Utiliser sa carte NOOBS 42
  - Créer une carte SD système 42
- Chapitre 3. Connecter votre Raspberry Pi 44
  - Insertion de la carte SD 47
  - Branchement du module caméra Raspberry Pi 49
    - Branchement de la caméra sur une Pi Zéro W 50
    - Branchement de la caméra sur les autres modèles 51
  - Adaptation des Pi Zéro et Pi Zéro W 51
  - Branchement d’un écran ou d’un téléviseur 52
    - Branchement d’un écran HDMI ou DVI 52
    - Branchement d’un téléviseur composite 53

Branchement d’une multiprise USB (hub) 54
Branchement d’un clavier et d’une souris 55
Branchements audio 55
Branchement au routeur 56
Branchement de l’alimentation 57
Démarrage 57
Ouverture de session (login) 64
Configuration du système Raspbian 65
Configuration Wi-Fi 69
Configuration Bluetooth 70
Test du module Caméra 71
Configurer la partition data 72
Prêt pour l’aventure ? 73
PARTIE 2. Découvrir Linux 75
Chapitre 4. L’environnement graphique 76
Naviguer sur le bureau 77
Découverte du menu général 77
Lancer un programme absent des menus 81
Retailler et fermer une fenêtre 81
Le Gestionnaire de tâches 83
Le Gestionnaire de fichiers 84
Navigation avec le Gestionnaire de fichiers 85
Copier et déplacer des fichiers et dossiers 90
Sélection multiple 90
Création d’un fichier vide ou d’un dossier 92
Suppression de fichiers et de dossiers 92
Mode d’affichage des fichiers 93
Ouvrir un dossier en root ou dans le terminal 94
Naviguer sur le Web 95
Le navigateur Web Chromium 97
Recherche dans une page Web 98
Navigation par onglets 98
Favoris et autres marque-pages 99
Protection de la vie privée 100
La messagerie Claws Mail 100
Le Visionneur d’images 101
L’éditeur de texte 105
Personnaliser le bureau graphique 106

Ajouter et enlever des applications 107
Sauvegarde des données 108
Déconnexion et arrêt système 109
Chapitre 5. L’interpréteur Linux shell 111
L’invite de commande 112
Exploration du système Linux 113
Affichage des fichiers et répertoires 113
Changement de répertoire 114
Contrôler les types de fichiers 114
Retour au répertoire de niveau supérieur 115
L’arborescence de répertoires 117
Chemins d’accès relatifs et absolus 121
Options de listage avancées 124
Droits d’accès ou permissions 128
Contrôler le défilement avec less 131
Optimiser la saisie de commandes 132
Créer un fichier par redirection (< et <<) 133
Conseils de nommage des fichiers sous Linux 135
Créer un répertoire (mkdir) 136
Supprimer un fichier (rm) 137
Sélection multiple avec les métacaractères * et ? 139
Supprimer un répertoire (rmdir, rm) 142
Copier et renommer des fichiers (cp) 143
Installer et gérer les logiciels du Raspberry Pi 145
Actualiser le cache 146
Trouver le nom d’un paquetage 146
Installer un logiciel 147
Lancer un logiciel 148
Mettre à jour un logiciel 148
Supprimer un logiciel et faire de la place 149
Lister les programmes installés (dpkg) 150
Gérer les comptes utilisateurs 151
Se documenter sur les commandes 153
Définir une nouvelle commande Linux 155
Arrêter et redémarrer le système 157
PARTIE 3. Travailler et se cultiver avec le Raspberry Pi 159
Chapitre 6. La bureautique avec le Raspberry Pi 160

Installation de LibreOffice sur le Raspberry Pi 161
Démarrage de LibreOffice 161
Sauvegarder le travail 162
Rédiger un courrier avec Writer 163
Gérer un budget avec Calc 165
Créer un diaporama avec Impress 169
Créer une invitation avec Draw 172
Chapitre 7. Retouches photos avec GIMP 176
Installer et démarrer GIMP 177
La fenêtre de travail de GIMP 177
Redimensionner une image 180
Rogner une photo 182
Rotation et miroir 184
Corriger les couleurs 184
Corriger les imperfections 185
Exporter une image dans différents formats 186
Pour en savoir plus au sujet de GIMP 187
Chapitre 8. Le Raspberry Pi comme centre multimédia 188
Préparation du centre multimédia 189
Découverte du centre multimédia 189
Ajouter des fichiers multimédias 191
Ajouter des fichiers audio 191
Ajouter des fichiers vidéo 194
Ajouter des photos 195
Exploiter des flux multimédias 196
Écouter de la musique 196
Visionner des vidéos 197
Visionner des photos 198
Paramétrage de LibreELEC 198
Utiliser une télécommande 199
Éteindre le centre multimédia 199
Le multimédia dans Raspbian 200
PARTIE 4. Apprendre à programmer avec le Raspberry Pi 202
Chapitre 9. Introduction à la programmation visuelle avec Scratch 203
Programmer, c’est quoi ? 204
Scratch 1 ou Scratch 2 ? 204
La fenêtre de l’atelier Scratch 206
Positionner et redimensionner un objet 208

Déplacer l’objet par programme 208
Mouvements et orientation 209
Les coordonnées dans la scène 211
Afficher les informations d’objet 214
Modifier l’aspect d’un objet 214
Utiliser des costumes 214
Utiliser les bulles Dire et Penser 216
Les effets graphiques 217
Redimensionner un objet 218
Contrôler la visibilité d’un objet 218
Effets sonores et musique 219
Créer un script 221
Apprendre le moonwalk 222
Ralentir un mouvement avec le bloc Attendre 223
Sauvegarder son travail 224
Nouveautés de Scratch 2 225
Chapitre 10. Programmer un jeu d’arcade en Scratch 227
Lancement d’un nouveau projet Scratch 228
Modifier l’arrière-plan 230
Ajouter des objets visuels 230
Dessiner un objet visuel dans Scratch 231
Nommer les objets 235
Contrôler l’exécution des scripts 236
Drapeau vert pour lancer les scripts 236
Le bloc Répéter indéfiniment 237
Contrôle d’un objet avec le clavier 239
Contrôler un objet avec un autre 240
Exploiter des valeurs aléatoires 243
Détecter une collision entre deux objets 244
Utilisation de variables 245
Déplacer automatiquement un objet 247
Détecter et corriger un bogue 249
Ajouter un script pour la scène 252
Dupliquer un objet 253
Tester le jeu 253
Régler la vitesse de jeu 254
Pour aller plus loin 254

Chapitre 11. Programmez en Python 256
Démarrer Python 257
Saisie de commandes Python 258
Quelques calculs en Python 260
Un programme de tables de multiplication 262
Rédiger et tester un premier programme Python 262
Utiliser des variables 264
Récupérer une saisie utilisateur 265
Afficher des mots, des variables et des chiffres ensemble 266
Répéter des instructions avec une boucle for 268
Création du projet Chatbot 272
Le concept de liste 273
Créer une liste de réponses aléatoires 277
La boucle de répétition while 280
Ajout d'une sous-boucle pour forcer la saisie 282
Les dictionnaires de données { } 283
Créer une nouvelle fonction 285
Fonction de balayage du dictionnaire 289
Création du sous-programme principal 292
Test de Chatbot 292
Code source complet de Chatbot 293
Chapitre 12. Un jeu avec Python et Pygame Zero 298
Récupérer des images et des sons 299
Préparer les dossiers 301
Écrire et exécuter un premier programme 302
Exécution d'un programme Pygame Zero 302
Détecter les clics de souris 306
Animer vos acteurs 308
Utiliser des nombres aléatoires 310
Ajouter d'autres acteurs 311
Regénérer des nuages 314
Cliquer sur plusieurs nuages 316
Ajouter un chronomètre 317
Ajuster la difficulté du jeu 318
Listing complet du code source 319
Pour aller plus loin dans Pygame Zero 321
Chapitre 13. Programmer Minecraft en Python 323

Découvrir le monde Minecraft 324
Se déplacer 325
Créer et casser des blocs 326

Préparer Python pour Minecraft 327
Le module Minecraft 327
Les coordonnées spatiales dans Minecraft 328
Positionner le joueur 330
Ajouter des blocs avec setBlock () 330
Interdire les modifications 332
Définir les paramètres du labyrinthe 333
Poser les fondations 335
Monter les murs 336
L’algorithme de création du labyrinthe 337
Définition des variables simples et des listes 339
Définition des fonctions 340
Le bloc de traitement principal 342
Ajouter un toit ? 345
Placer le joueur pour commencer 345
Le code source complet 346
Modification du programme 350
Chapitre 14. De la musique avec Sonic Pi 352
L’interface visuelle de Sonic Pi 353
Vos premières notes 355
Noms des notes et accords 357
Pour alléger l’écriture 358
Une mélodie aléatoire avec Shuffle 359
Au gré du hasard 360
Nommer vos listes 361
Faire jouer des hauteurs au hasard 361
Le temps réel avec live_loop 362
Reproduire des échantillons 364
Ajouter des effets spéciaux (Fx) 366
Synchronisation avec une piste rythmique 366
Récapitulons 367
Plus loin avec Sonic Pi 370
PARTIE 5. Découvrir l’électronique avec le Raspberry Pi 371
Chapitre 15. Principes des circuits électroniques et soudure 372

Qu’est-ce qu’un circuit électrique ? 373
Nature de l’électricité 373
De la théorie à la pratique 376
Représentation des circuits électroniques 379
Calcul des valeurs dans un circuit 379
Contraintes techniques des composants 381
Test d’un circuit avec un simulateur 382
Le connecteur GPIO 382
Usage général ou spécifique 383
Étude d’une broche GPIO 384
Utilisation pratique d’une broche de sortie 385
Utilisation d’une broche GPIO en entrée 387
Découverte du fer à souder 389
Vos premières soudures 391
Cartes d’extensions du commerce 392
La carte Gertboard 393
La carte Pi Face 394
Autres cartes d’extension 395
Quelques cartes d’extension 395
La carte Sense HAT 396
La carte Skywriter HAT 397
La carte Xtrinsic Sense Board 398
Autres cartes 399
Chapitre 16. Contrôler les circuits du RasPi 400
Accès aux broches GPIO 400
Soudage du connecteur sur le Pi Zéro 403
Les connecteurs d’extension 404
Réaliser les connexions 406
Votre premier circuit 406
Faire clignoter la diode LED 408
Avec Scratch 1.4 408
Contrôler la vitesse de clignotement 410
Clignotement de LED avec Python 413
La librairie GP Zéro 416
Construire un dé électronique 420
Alea jacta est ! 421
Test du câblage 423

Générer le nombre 424
Afficher le nombre 424
Pour aller plus loin 430
Simulateur de passage piéton 431
Montage du circuit de passage piéton 436
Le logiciel du passage piéton 437
Chapitre 17. Utiliser des diodes LEDRGB 444
Des LED de toutes les couleurs 444
Mélangeons les couleurs 446
Lumière diffuse ou non 447
Une vraie palette de couleurs 447
Des LED intelligentes 450
Émulation du protocole APA102 454
Définition d’une classe 455
Les sept envahisseurs de l’arc-en-ciel 460
Le pied jongleur Keepy Uppy 468
Des galaxies de LED 471
Limitations en courant 472
Discrimination des états logiques 473
Fréquence de rafraîchissement 474
Autres modèles de guirlandes LED 475
Rubans de LED 476
Matrices de LED 476
Chapitre 18. L’identification par radiofréquences 484
Principe du RFID 484
Structure mémoire d’une carte MIFARE 490
Un premier juke-box RFID 493
Un juke-box RFID plus versatile 496
Pour aller plus loin 500
PARTIE 6. Les Dix Commandements 502
Chapitre 19. Dix logiciels sélectionnés pour le Raspberry Pi 503
Penguins Puzzle 503
FocusWriter 504
Mathematica 505
Invaders 3D 507
Fraqtive 507
Tux Paint 508

Grisbi 509
Beneath a Steel Sky 510
Sense HAT Emulator 511
Brain Party 512
Chapitre 20. Dix projets enthousiasmants 514
Un lecteur d’audiolivre à bouton unique 514
Une station météo 515
Un cardiogramme 515
Un skateboard électrique 516
Un canon à T-shirts 516
Panflute hero 517
Magic Mirror 517
Pi in the sky 518
Le Turc mécanique 519
Sound Fighter 520
Annexe A. Configurer et dépanner votre Raspberry Pi 521
Dépanner le Raspberry Pi 522
Ajuster les paramètres système 526
L’éditeur Nano pour modifier config.txt 528
Dépannage de l’affichage 530
Ajustement des dimensions de l’écran 534
Autres paramètres 534
Correction des problèmes de son 534
Montage manuel de périphériques amovibles 535
Correction d’erreurs d’installation de logiciels 537
Dépannage de la connexion réseau 537
Connexion sécurisée SSH 538
Sommaire 541